D8.1 Specification of Security and Privacy Handling	
Document ID:	D8.1
Document version:	1.1
Document status:	Final
Dissemination level:	PU
Deliverable number:	D8.1
Deliverable title:	Specification of security and privacy handling
WP number:	WP8
Lead beneficiary:	University of Agder, Norway
Main author(s):	Nils Ulltveit-Moe, Terje Gjøsæter, László Erdődi, Stefan Siegl
Nature of deliverable:	R
Delivery date from Annex 1:	28/02/2015
Actual delivery date: (of the revised version)	10/07/2015
Funding scheme / call:	STREP-FP7-ICT-2013-11
Project / GA number:	619560
Project full title:	Scalable Energy Management Infrastructure for Aggregation of Households
Project start date:	01/03/2014
Project duration:	36 months

Funded by the
European Union

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 619560.

Executive Summary

This document contains high-level privacy and security requirements for the SEMIAH home energy management system. It does an investigation of state of the art and current research including an overview of relevant standards. This includes describing the considerations that are necessary to achieve security by design, including secure design patterns and design principles. In a similar way it describes the necessary considerations for achieving privacy by design based on the seven foundational privacy-by-design principles.

Section 2 of the document elaborates on methods and tools for security and privacy management for SEMIAH, including tests and risk metrics for identifying gaps in security and privacy/confidentiality, as well as a discussion on possible safeguards and vulnerability management processes that can be used to mitigate the identified gap in security. This includes mapping the security measures for smart grids identified by the ENISA smart grid task force into appropriate security measures in SEMIAH. Example security measures are how to perform risk assessment, information leakage detection and controlling security in the home gateway.

Section 3 describes how security and privacy can be enhanced for SEMIAH when running in a cloud-based environment, including considerations of security handling in message oriented middleware, as well as service authentication and authorisation and cryptographic methods for securing the Demand/Response protocol. The section also describes how software security can be achieved using good practices such as secure design patterns and tools and techniques for security testing.

Section 4 goes into detail on how privacy and security assessment and security monitoring of the SEMIAH demonstrators will be performed using techniques such as intrusion detection and prevention systems, in order to increase the security awareness in the system, as well as having the ability to automatically respond to certain threats. The section also describes how vulnerability testing and privacy assessments can be performed as well as discussing how the risk assessment needs differ between requirements for the SEMIAH Pilots and potential full scale rollout of the SEMIAH Demand/Response system.

Abbreviations

API	Application Programming Interface
CRE	Common Remediation Enumeration
CVE	Common Vulnerabilities and Exposures
CWE	Common Weaknesses Enumeration
D	Deliverable
DER	Distributed Energy Resources
DOS	Denial of Service
DDOS	Distributed Denial of Service
DOW	Description of Work
DSO	Distributed System Operator
EC	European Commission
ENISA	European Union Agency for Network and Information Security
HEMG	Home Energy Management Gateway
HEMS	Home Energy Management System
IaaS	Infrastructure as a Service
IDS	Intrusion Detection System
IEEE	Institute for Electrical and Electronics Engineers
JSON	JavaScript Object Notation
MAC	Media Access Control
NIST	National Institute of Standards and Technology
PaaS	Platform as a Service
PRECYSE	Prevention, protection and Reaction to Cyber attackS to critical InfrastructurEs
SaaS	Software as a Service
SAML	Security Assertion Markup Language
SCADA	Supervisory control and data acquisition
SIEM	Security Information and Event Management
SQL	Standard Query Language
TSO	Transmission System Operator
URL	Uniform Resource Locator
OGEMA	Open Gateway Energy Management
XML	Extensible Markup Language
XACML	Extensible Access Control Markup Language
VPP	Virtual Power Plant
WP	Work Package
WT	Work Task

Contents

List of Figures

List of Tables

1 Introduction

WP8 is transversal to the entire work plan of SEMIAH. This means that security and privacy functions should be integrated in all parts:

- Front-end (OGEMA)
- Mobile user interface
- Back-end/aggregator function.

An objective is to implement methodologies, techniques and tools that make the energy management gateway and Demand/Response service resilient against cyber-attacks. Attack and fraud attempts should be detected and reported and the SEMIAH infrastructure should be safe, secure and respect privacy/confidentiality.

D8.1 is mainly covered by task T8.1, which aims at defining the smart grid security and privacy handling for the Demand/Response service. This includes defining security requirements as well as supporting the misuse cases and misuse scenarios defined in D3.1. It also contains a functional specification for the privacy and security management methodology as well as supporting tools and techniques that may be built into SEMIAH or support the Demand/Response service. This is reflected in the main business requirement for Security and Privacy based on Description of Work (DoW), which is presented in Table 1 below.

This specification is at a relatively high abstraction level. A more practical system architecture, focusing on the stakeholders, data and interface to other systems will be covered by D8.2.

Table 1 Business Requirements for Security and Privacy.

AB 6	The consortium will integrate security and privacy functions to prevent that the system cannot be compromised.	AB6. 1	The project shall integrate security functions to prevent that the system cannot be compromised.
		AB6. 2	The project shall integrate privacy functions to prevent that the privacy and integrity of the system users is not compromised.

1.1 WP8 Objectives

The objectives of WP8 is to identify necessary privacy and security requirements to ensure safe and secure operation with only leakage of necessary personally identifiable information, for example for billing purposes. WP8 aims at implementing privacy and security by design, which means that privacy and security requirements are being designed into the technical solution from the start.

Another objective is to define a supporting privacy and security management process for mitigating privacy and security risks. Finally, WP8 aims at verifying that the implemented privacy and security controls work as expected in the system demonstrator. This means being able to detect and mitigate system weaknesses and attacks.

2 State of the art and current research

This section gives an overview of relevant standards, and investigates necessary considerations in order to implement Security by Design, including secure design patterns and principles, secure authentication, data and control segregation and trust, input data validation and finally correct use of cryptography. The Privacy by Design principles are also being discussed, as well as identification of sensitive information with a focus on the user's perspective. We also discuss the protection scheme which will be based on existing and improved tools and countermeasures for attack detection, risk analysis, vulnerability analysis and privacy enforcement. These will amongst others be based on and extend tools, methods and techniques developed for the PRECYSE FP7 project for ensuring network security, as well as using other relevant security tools and methods for verifying software security.

2.1 Overview of Relevant Standards

ISO/IEC 27001:2013 describes security techniques for information security management systems. It amongst others provides a model for establishing, implementing, operating, monitoring, reviewing, maintaining and improving an information security management system. It uses a top-down, risk-based approach which is technology-neutral.

ISO/IEC 27002:2013 describes security techniques in the form of a code of practice for information security management. It establishes guidelines and general principles for initiating, implementing, maintaining, and improving information security management within an organization. The standard also provides guidelines for the development of effective security management practices to help build inter-organizational activities.

ISO/IEC ISO27019:2013 describes information security management guidelines based on ISO/IEC for process control systems specific to the energy utility industry. This standard is of high importance for SEMIAH.

ISO/IEC ISO15408:2209 describes security techniques in the form of evaluation criteria for IT security (Common Criteria). ISO 15408 establishes the general concepts and principles of IT security evaluation and specifies the general model of evaluation of security properties of IT products. Common Criteria contains important concepts to keep it in mind.

ISA/IEC-62443 covers industrial network and system security.

ISA/IEC-62443 is a series of standards, technical reports, and related information that define procedures for implementing electronically secure Industrial Automation and Control Systems.

From the Internet Engineering Task Force (IETF) documentations the following documents are relevant:

RFC 2196: Site Security Handbook

RFC 2818: HTTP over TLS

There are other recommendations that should be considered for the SEMIAH project. The CERT Oracle Coding Standards for Java are very important for writing secure applications for the OGEMA gateway [1].

The US National Institute of Standards and Technology (NIST) has relevant standards. Since they are not European standards, they may be considered only as an extension. The 800 series of NIST are of general interest to the computer security community. This series reports on ITL's research, guidelines, and outreach efforts in computer security. Of particular interest are:

NIST 800-12 An Introduction to Computer Security NIST 800-14 Generally Accepted Principles and Practices for Securing Information Technology Systems NIST 800-26 Security Self-Assessment Guide for Information Technology Systems.

IEC 62443 “Industrial Automation and Control Systems Security” and ISO/IEC TR 27019 “Information security management guidelines based on ISO/IEC 27002 for process control systems specific to the energy utility industry” can also be relevant for SEMIAH.

The Smart Energy Profile 2.0 (SEP2)[1] is also very relevant for the SEMIAH demand/response protocol.

The project will benefit from open source Information Security Management tools from the PRECYSE project[2], which have been adapted to the freely available Spanish MAGERIT risk assessment standard [2], [3], as well as being inspired by for several of the above mentioned standards, for example the ISO27000 set of standards.

2.2 Consideration of Security by Design

The goal of a security by design is to ensure that the security of the system does not break because of design flaws. This can be achieved by designing a system that supports and enforces the necessary authentication, authorisation, confidentiality, data integrity, accountability, availability and non-repudiation requirements, even when the system is under attack [4].

2.2.1 Secure Design Patterns

Security needs to be a part of the development lifecycle, to build a secure system in a systematic way. Security requirements are identified during requirements analysis. In order to find security requirements we can for example identify abuse cases, do conformance test (e.g. ISO27001) or perform a threat modelling. In this section we will focus on guidelines from NIST (NIST IR 7628 [14]) to derive some security requirements for the SEMIAH backend-system. Those security requirements serve as input for the design stage. There exist some recurring problems in software design, which are solved by using design patterns. A design pattern in general describes a reusable solution to common problems in software design. A secure design pattern in particular is a description or template of how to solve a security problem. We want to identify those secure design patterns that help us to fulfil our collected security requirements.

We describe more in detail how secure design patterns will be applied in Section 4.6.

2.2.2 Secure Design Principles

The IEEE Center for Secure Design published a guide on how to avoid the top ten software security design flaws in 2014 [4]. This guide proposes a set of ten general security requirements for achieving security by design:

- Earn or give, but never assume trust;
- Use an authentication mechanism that cannot be bypassed or tampered with;
- Authorize after you authenticate;

[1] SEP2 protocol: http://www.grid2home.com/iot-technologies/smart-energy-2-0
[2] Prevention and Reaction to Cyber-attacks to Critical Infrastructures (PRECYSE) http://www.precyse.eu

- Strictly separate data and control instructions, and never process control instructions received from untrusted sources;
- Define an approach that ensures all data are explicitly validated;
- Use cryptography correctly;
- Identify sensitive data and how they should be handled;
- Always consider the users;
- Understand how integrating external components changes your attack surface;
- Be flexible when considering future changes to objects and actors;

This document distinguishes between bugs, which are implementation level software problems, and flaws, which is a problem at a deeper level. A flaw is the result of a mistake or oversight at design level [4].

The remaining part of Section 3 is based on the security requirements and good practices for avoiding the top ten software design flaws [4], adapted to the business case of SEMIAH.

Software systems consisting of more than one component rely on the composition and cooperation of these components to successfully accomplish their purpose. These designs often depend on the correct functioning of the existing parts. They will be inherently insecure if any of these parts are run in a potentially hostile environment, for example in a mobile device or in cloud-based services in SEMIAH's case.

Offloading security functions from server to client exposes those functions to a much less trustworthy environment, which is one of the most common causes of security failures predicated on misplaced trust.

Designs that place authorisation, access control enforcement of security policy or embedded sensitive data in client software, thinking that it will not be discovered, modified or exposed by adversaries are inherently weak. Such design will often lead to compromises. For SEMIAH, this for example means that we should not trust pure client side access control decisions, for example using hard-coded user name and password in the home energy management gateway. The system should rather rely on a centrally managed federative access control system, using existing standards. The mobile device should also use a centrally managed access control solution for accessing the web server. Also calls into your APIs from business partners (e.g. from the market side, DSO or TSO) should be considered client software which requires proper access control enforcement.

When untrusted clients send data to your system or perform a computation on its behalf, the data sent must be assumed to be compromised until otherwise proven. Such systems are therefore unsuitable for performing security sensitive tasks. In the case of SEMIAH, there will be limits for how much we can trust the underlying cloud-based platform. We should therefore be very cautious on how computations and data that may be sensitive are being treated in this case. We should also aim at building the home gateway secure by design, so that it by default can be trusted by the rest of the system.

An important principle is that all data received from an untrusted client are properly validated before processing [4].

Common pitfalls we should avoid are [4]:

- Do not make assumption on ordering of API calls (e.g. in our case make REST calls idempotent where possible, to make the code independent of server side state.)
- Do not assume that the user interface is able to restrict what the user can send to the server (i.e., perform boundary control checks, use typed SQL clauses instead of strings etc.).
- Avoid building business logic solely on the client side, or attempting to store a secret in the client.

- Do not assume that intellectual property sent to the client can be protected using technical means. If it still is necessary to do this, then use data protection mechanisms to delay the leakage of sensitive material, e.g. obfuscation or anti-debugging.

If private or confidential information must be stored or sent to the client, the system should be designed to be able to cope with potential compromise, i.e. the sensitive information should not be revealed. In particular aim at avoiding the following pitfalls:

- The same shared secret should not be used on all the clients, use different shared secrets.
- Make the validity of what is stored on the client limited in time, for example using time limited cryptography or similar techniques to set an expiration date for data stored in the client, watermark intellectual property, and verify that computations that are privacy or security sensitive are being performed correctly.
- Design the system to degrade gracefully if one or a set of clients have been compromised.
- Make sure all data are properly validated before processing.
- Consider the context where code will be executed, where data will go and where data comes from, in order to avoid vulnerabilities due to trusting components that are not trustworthy.

2.2.3 Use an authentication mechanism that cannot be bypassed or tampered with

One goal of a secure design is to prevent an entity (user, service etc.) from gaining access without first authenticating. Once authenticated, a securely designed system should prevent that the user changes identity without re-authentication [4] .

Authentication techniques require one or more factors, such as:

- Something you know (e.g. password)
- Something you are (e.g. biometrics)
- Or something you have (e.g. a smartphone, or cryptographic key)

Multifactor authentication is the technique of require multiple distinct factors to prove your identity. Authentication using a cookie on a web browser may be sufficient for some noncritical functions. For sensitive functions in SEMIAH, stronger two-factor authentication should be used (for example a combination of encryption key and password or SMS one time key + password). An example of functions that should require two-factor authentication in SEMIAH are functions for:

- Managing user account (e.g. reset password)
- System administration and privileged accounts
- Managing sensitive data or functions

The SEMIAH system must consider the strength of the authentication when deciding which services the user is authorised for. This applies both for human-computer interaction and interaction between computer-based services in SEMIAH.

Authentication must be mandatory for SEMIAH to avoid risking that an authorised entity gets access to systems or services it should not have access to. IP and MAC addressees must not be used as substitute for authentication, since these can be spoofed.

When updating passwords, the user must perform a full re-identification using the required authentication procedures (e.g. present existing password and other factors that are required for authentication).

Authentication should result in the creation of a token, capability or ticket representing a user or service that is used throughout the session between the authorised parties. These credentials must be created so that they are not easy to forge, so that an attacker cannot easily bypass this authentication mechanism. Upon successful authentication, the user may be provided with an authentication credential, token or ticket which can be provided back to the system, so that the user does not need to be re-authenticated for every request or transaction. The credential, token or ticket is confidential information that must be protected against disclosure, which means that services requiring authentication should be encrypted if these services run over an untrusted network.

The authentication system must impose a time limit for how long an authentication session is valid. After this time limit, the entities need to re-authenticate. Re-authentication should also be performed after a period of inactivity or prior to critical operations. The re-authentication scheme must be designed to be practical and usable, meaning that a frequent re-authentication scheme should not be necessary for noncritical services that the user frequently uses.

SEMIAH must ensure that passwords are stored securely, using existing best practices within cryptography. SEMIAH should also reuse existing password management systems where possible, instead of building new ones.

SEMIAH should use a single method, component or system responsible for authenticating users or services (e.g. a single-sign-on system), instead of relying on device-local authentication.

Authentication mechanisms are critical to secure design, and may be susceptible to various forms of tampering and may be bypassed if not designed correctly [4]. SEMIAH should therefore use a single authentication mechanism that requires two authentication factors for critical services, and at least one factor for noncritical services. The authentication credentials must be unforgivable and stored so that it cannot easily be reverted if the stored form is stolen.

2.2.4 Authorise after you authenticate

Authorisation of the user must be performed after successful authentication. Authorisation depends not only on the privileges associated with an authorised user, but also on the context of the request. This context may also include the time and location of the request.

The authorisation system must support that authorised personnel can modify or revoke a user's authorisation, in order to avoid vulnerabilities due to out-of-date authorisations (e.g. when employee leaves company).

Sensitive operations may require (re)authentication in order to perform authorisation. Some sensitive operations (e.g. critical system configurations or updates) may also require two (or more) people or entities to authorise critical transactions. Such operations will require two-factor authentication, to ensure that one user cannot impose as both users. SEMIAH should aim at using a common infrastructure for performing authorisation checks.

2.2.5 Strictly separate data and control instructions, and never process control instructions received from untrusted sources

Mixing data and control instructions in a single entity, especially a string, can lead to injection vulnerabilities [4]. This means that the lack of separation between data and code can lead to untrusted data controlling the execution flow of a software system. Examples of such vulnerabilities that happen at low level are memory corruption vulnerabilities, e.g. pointer handling in C causing buffer overflow vulnerabilities. At higher level, co-mingling of control and data often occurs when interpreting domain-specific languages. For example SQL query injection, cross-site JavaScript injection and shell command injection.

SEMIAH should consider control-flow integrity and segregation of control and potentially untrusted data as a design goal. This can be done by avoiding to expose methods or endpoints that consume strings which embed both control and data. Prefer instead to expose methods or endpoints that consume structured types that impose strict segregation between data and control information. When using existing APIs, avoid using APIs that mingle data and control information in their parameters. For example, use a typed SQL interface passing typed SQL parameters instead of merging SQL using concatenated strings based on potentially untrusted data. If data separation is not possible, then try to encapsulate injection prone interface through a higher level interface that enforces strict segregation between control statements and potentially untrusted data.

Designs that rely on the ability to transform data into code should take special care to validate the data as fully as possible, and to contain the set of computations that can be performed using data as an input language. Specific concerns are for example the eval function, query languages and exposed reflection. Code generation is for example safer than using reflection for serialising/de-serialising objects.

2.2.6 Define an approach that ensures all data are explicitly validated

Software systems and components commonly make assumptions about data they operate on. It is important to explicitly make sure that such assumptions hold. Vulnerabilities frequently arise from implicit assumptions about data which can be exploited if an attacker subverts and invalidates these assumptions.

SEMIAH should aim at ensuring that comprehensive data validation takes placed, and that all assumptions about data have been validated when they are used. This should be done in a way that makes it feasible for WP8 to verify using code reviews. This means designing or using centralised validation mechanisms to ensure that all data entering the SEMIAH system are appropriately validated.

Elements that should be considered are:

- Web applications should use a request filter and interceptor to intercept all incoming requests and apply basic input validation to all request parameters.
- Communication protocols should validate all fields of all received messages before any processing takes place.
- XML parsers should validate the input document.
- Transform data into a canonical form before performing syntactic and semantic validation, if possible. This ensures that validation cannot be bypassed by supplying input in a different encoding.
- Use common libraries for validating primitives (URLs, email addresses etc.).
- Validation should be based on a whitelisting rather than a blacklisting approach.
- If the protocol is state-based, make sure the input validation is state-aware.
- Explicitly revalidate assumptions of nearby code that relies on them, for example, SEMIAH's business logic should explicitly restate and check as preconditions all assumptions that it relies on. Liberal use of precondition checks (e.g. assertion clauses) in the entry points of software modules and components is highly recommended. This also allows for local reasoning about the correctness of a component, since assumptions are explicitly checked and stated.
- Use implementation-language-level types to capture assumptions about data validity, for example validate that date and time strings are well-formed etc.

Failure to address this security design principle can cause the following problems:

- Injection vulnerabilities if untrusted data are used without validation;
- File path traversal vulnerabilities, unless path components (path separators) are validated;
- Externally controlled string within a web document can cause cross-site-scripting vulnerabilities;
- Attempting to validate data not in its canonical form can cause input validation to fail;
- Lack of input validation for non-type-safe languages like C can cause memory corruption vulnerabilities;
- Accepting input from untrusted sources without an upper data size bound can cause resource exhaustion;
- Invalidated data are potentially more harmful if they can influence control flow.
- These vulnerabilities may cause state transitions that the programmer did not intend.

State-dependent input validation goes a long way in reducing the risk of these vulnerabilities.

2.2.7 Use cryptography correctly

Cryptography is one of the most important tools for building secure systems [4].| Proper use of cryptography can ensure privacy and confidentiality of data, protection from unauthorised modification (integrity protection) and for authenticating the source of data. However, getting cryptography right is hard. Some common pitfalls that SEMIAH should avoid, if possible, are:

- Avoid making your own cryptographic algorithms or implementations. Use instead standard algorithms and libraries.
- Avoid misuse of libraries and algorithms. There may for example be specific security requirements for an encryption scheme – it may for example protect the confidentiality of data, but not against malicious modifications. There may also be security requirements for how an initialisation vector should be chosen.
- Avoid poor key management, for example using hard-coded keys in embedded devices and application software, failure to allow key revocation and/or rotation, avoid using weak cryptographic keys (too short or predictable) and weak key distribution mechanisms.
- Avoid using poor pseudo random number generators, since cryptographic operations assume that random numbers have strong properties. Avoid re-using the random numbers.
- Failure to centralise cryptography, for example where different teams implement their own cryptographic routines. Try instead to get it right once, and reuse it everywhere.
- Failure to allow for adaptation and evolution of cryptographic algorithms. Make the design instead future-proof by supporting upgrading cryptographic algorithms and tool versions.

Cryptography is hard to get right. Always work with an expert, or have peer-review by experts on chosen solutions. Crypto experts should provide an API abstraction around a strong cryptographic library, so that developers are not making decisions on algorithms and cipher modes, and also so that algorithms can be changed if necessary.

2.2.8 Identify sensitive data and how they should be handled

One of the first tasks that should be done during design of SEMIAH is to identify sensitive data and determine how to protect it appropriately. Data sensitivity may depend on many factors, including regulatory requirements, company policies of SEMIAH partners, contractual obligations and user expectations. Sensitive data may be data from users, sensors, cryptographic material and personally identifiable information. Creating a policy that explicitly identifies different levels of classification should be done as a first step.

Regulatory requirements include requirements on data protection from relevant smart-grid standards and also general legal requirements like the EU data protection directive and communication directive [5], [6].

Different data may require different data protection measures. For some data, confidentiality is critical. For SEMIAH this may for example be market sensitive information or personally identifiable information. For other data, availability will be critical, for example the required data to ensure that the backend aggregation algorithm can operate or data required to maintain grid stability. There will also be data where data integrity is most important, for example for communication between the SEMIAH backend and frontend, in order to ensure that commands and responses cannot easily be forged, spoofed or manipulated.

Technical controls that can be applied to sensitive data include access control mechanisms (including file protection, memory protection and database protection mechanisms), cryptography (e.g. the Reversible anonymiser provided by UIA) to protect data confidentiality or integrity and redundancy and backups to preserver data availability.

SEMIAH needs to identify trust boundaries for data in transit and protect these boundaries with appropriate data protection policies. There will for example be trust enclaves in SEMIAH for major functional blocks like the frontend, backend and web server.

Policy requirements and data sensitivity can change over time, which means that a privacy and security management process is needed to revisit and revise the data protection policies and their design implementations.

2.2.9 Always consider the users

The SEMIAH demand response system will interact with different types of users and stakeholders, ranging from technicians, system administrators, house owners, DSO and TSO managers and energy traders. The security stance of SEMIAH is linked to what the users do with it. It is therefore important that all security related mechanisms are designed in a way that makes them easy the deploy, configure, use and securely update. Security, just like privacy, is not a feature that can be easily bolted on to a software system. It is rather a property emerging from how SEMIAH has been built and operated.

SEMIAH needs to consider how the physical abilities, cultural biases, habits and idiosyncrasies of the intended users influence its overall security. Some users may for example discover capabilities outside the intentions of the system design. Some of these capabilities may have significant security implications. Usability and user experience considerations are important factors for ensuring that the software operates in a safe and secure manner. The systems should be designed with easy to use interfaces with sufficient, but not excessive functionality. Using appropriate security controls is crucial.

SEMIAH should neither assume that the intended user is interested in security nor that the user is well meaning. The design should facilitate secure configuration and use by authorised parties and prevent or mitigate abuse by adversaries aiming at weakening or compromising the system.

Failing to address this design principle can lead to:

- Privilege escalation from failure to implement an authorisation model which is sufficiently tied to the authenticated user. It can also happen if higher-privileged functions (e.g. root/sysadmin) are not being protected by the authorisation model and where assumptions about inaccessibility are incorrect.
- Failure of appropriate authorisation can allow a breach of authorisation and isolation between users, such that one can access others data.
- Inadvertent disclosure by the user may happen if the authorisation model is difficult to understand.
- Configurations that by default are open (e.g. a default PERMIT firewall policy), e.g. on system configuration or first run assume that the user understands the implications of this. It is better to have a reasonable level of security even during initial configuration.
- If the security configuration is difficult or non-intuitive, then it will be difficult to configure the product to conform to the required security policy.
- The system should consider the disk that authenticated and properly authorised users also can be attackers, essentially giving authorised users opportunities to misuse the system. SEMIAH can for example mitigate this by monitoring and logging system critical operations that authorised users may do, in order to detect attacks by insiders.
- The security of SEMIAH must be reasonable for the intended set of users. If the security makes the system too hard to use, then this could cause a risk for the business case of SEMIAH, since users may give up using it. This could also cause a risk that the system would not be configured properly, if the security functions are not manageable for the system administrators or technicians installing or managing the system.
- Application programmers interfaces must be sufficiently intuitive to use that the intended API can be understood and used correctly.

- Also consider the risk of collateral damage that can occur from software or data that is being included into the SEMIAH system. For example the risk of identifying privacy sensitive data on device usage.
- Make sure a user’s data is considered during setup, use and revocation/termination, both to ensure that data is not gathered/stored against the users’ wishes, or that SEMIAH holds on to data that should have been removed completely after the user stopped using the service and closed the account. It should for example be easy for the user to destroy private data when the user decides to stop the service. The user should be in control of his/her own data.
- SEMIAH should consider principles such as universal design, to avoid excluding classes of users from using the software or making the software too difficult to use effectively. Security functions should be usable also for users with different capabilities.

In other words: always consider the users and any other stakeholders in the design and evaluation of systems. Ensure that SEMIAH has a reasonable compromise between privacy, usability and security for home users and other stakeholders that will use SEMIAH. Security relevant decisions should only be taken by relevant, authorised stakeholders. Furthermore, it should also be an objective that the most common usage scenarios are made secure, using the secure by default principle. If users should be allowed to change any of these security settings, then it should be easy for them to find this option.

The security design must consider the risk of user fatigue, i.e. that the user clicks OK every time an application needs a specific permission, and try to design a system that avoids user fatigue while still providing an acceptable level of security and privacy to the user.

The objective should be to design SEMIAH as a system that is both secure and usable, a system that will be adopted on a wide scale and that is compatible with the values of users and other people impacted by them.

2.2.10 Understanding how integrating external components changes your attack surface

SEMAH will be based on several existing pieces of software. One such example, is the open source energy management gateway software OGEMA, however SEMIAH will probably be based on integrating many existing software systems, libraries, databases etc. Different parts of the SEMIAH system may also be produced by different development teams within the consortium.

It is important to make sure that the software components being integrated into SEMIAH have been tried and tested to verify that they stand up to the security requirements of SEMIAH, since the system will inherit the security weaknesses, limitations, maintenance responsibility and threat model of software or libraries we are including. This may cause a security gap which somehow must be solved, mitigated or accounted for before the system can be deployed.

It is amongst others important to consider the following factors [4]:

- How does each external component change the threat model of the entire system?
- Does it add to the attack surface?
- Does it modify entry points in the system that already has been considered in its own threat model?
- Were new features, capabilities or interfaces added, even though you are not using them? Can these features be disabled?
- Does the external component also include other external components with their own security weaknesses?
- Is the external component obtained from a known, trustworthy source?
- Does the external component provide security documentation that helps you better understand its threat model and the security implications of its configuration?

We should assume that incoming external components are not trustworthy until appropriate security controls have been applied, in order to align the component's attack surface and security policy with the ones that meet the requirements of SEMIAH. Examples of potential security issues with third-party components include:

- Using a library with known vulnerabilities identified in the Common Weakness Enumeration (CWE) or Common Vulnerabilities and Exposures (CVE) databases by Mitre;
- Including a library with extra features that entails security risks;
- Reusing a library that no longer meets current software security standards;
- Using a third-party service and hoping thereby to pass responsibility of security onto that service (in SEMIAH this is in particular relevant for cloud-based services);
- Configuration mistakes in the security of a library, e.g. using insecure defaults;
- Libraries making outbound requests to the maker's site or some partners of theirs;
- Libraries making inbound requests from some external source;
- A single external component including other components, causing multiple levels of inclusion using recursion (may apply to several Maven based libraries);
- Including pieces of functionality that offer unknown interfaces into the system, for example a command line interface for configuration of an included service, a panel or admin mode for a web component, hardcoded credentials for an authentication/authorisation module, debugging interface or backdoor or similar.

At a minimum, consider the following security requirements:

- Isolate external components as much as your required functionality permits;
  - o Use e.g. containers, sandboxes and drop privileges before entering uncontrolled mode;
- When possible, configure external components to enable only the functionality that is needed;
- If functionality that is not needed is included, consider how this changes the security posture (attack surface, risks, threats etc.) and therefore increases the security that must be implemented to counter the change;
- If it is not possible to configure the security properties according to the security requirements of SEMIAH, attempt to find another library or document that we are willing to accept the given risk;
- Validate the integrity and provenance of external components using cryptographically trusted hashes and signature, code signing artefacts and verification of the downloaded source. If no integrity mechanism is available, consider maintaining a local mirror of the library's source;
- Understand the risk of dynamically including components such as JavaScript from external sources. If the external host is compromised, you may be including attacker-controllet JavaScript;
- Follow sources that track of publish security related information regarding the external components being used, e.g. bug repositories, security focused mailing lists, CVE databases etc.;
- Make sure that the WP8 security team is aware of all external components used, so that these can be included in the threat intelligence collection effort;
- Maintain an up-to-date inventory of consumed external components and verify that it matches the versions included in SEMIAH, as well as that those are the latest know-secure versions available for each external component;
- Maintain a healthy distrust of external components:
  - o Whenever possible, authenticate the data flow between you system and external components;
  - o Consider all data coming from an external component to be tainted, unless proven valid (see section on data validation);
  - o Be sure to understand and verify the default configuration of the external component. For example, when including a crypto library, understand what values are used by default, understand entropy sources, algorithms and key lengths;

    - When using a component like a web server, understand its defaults concerning admin modes, ports it will be listening to and assumptions concerning how it interfaces with the operating system and with your own software;
- Document security relevant changes, for example:
    - Why a default is changed;
    - How external components are selected and the verification of it;
    - Security relevant assumptions made about it;
- This makes it easier to integrate new versions of the component or when considering alternative modules.
- When changing build defaults of external components, configuration options for deployment or source code, automate the procedure using the version control system or a patch file, then include the automated procedure in your build process, i.e:
    - Bring in the pristine component;
    - Apply your modifications;
    - Apply the self-tests of included tools, if applicable, to verify its configuration;
    - Use it for your build;
- This helps to maintain consistency between builds. It also makes it easier to know when your modifications need adjustments due to a version change in the external component.
- Design for flexibility. Sometimes an external component becomes too risky, or development is abandoned, or functionality is surpassed by another external component. In those cases, design the system so that external components easily can be replaced.

2.2.11 Be flexible when considering future changes to objects and actors

Software security must be designed for change, rather than being fragile, brittle and static [4]. The objective of SEMIAH is to fulfil a set of functional and security requirements, as set out in D3.2 and D8.1. However, the SEMIAH software and its running environment with related threats and attacks will change over time. This means that a security management process is needed, which considers the security implications of future changes and threats to the components.

Change management will amongst others need to consider configuration changes, enabling or disabling features and perhaps also dynamic loading of objects. This means that different variations of states will need to be tested to maintain the security posture of SEMIAH. There will also be changes at deployment when access control, permissions and other security related activities and decisions need to take place. SEMIAH will furthermore require continuous integration of energy management gateways and possibly other services, which creates a requirement for security flexibility.

The security management process also needs to handle the security erosion which occurs because threats change over time and system changes and new features may introduce new threats and vulnerabilities. Efficient system and configuration management is very important for SEMIAH, since it will need to be able to scale to potentially millions of home energy management gateways. This means that secure design must keep flexibility in mind. The following subsections elaborate on what this means.

2.2.11.1 Design for secure updates

It is easier to upgrade small pieces of a system than huge blocks. Smaller updates also ensure that the security implications of the update is better understood and controlled. The integrity and provenance of upgraded packages should also be verified. Make use of code signing and signed manifests to ensure that the system only considers patches and updates of trusted origin. Finally consider the maintenance burden on administrative personnel. As complexity increases, there is an increasing likelihood of making mistakes. This can in many cases be controlled by automating the installation process, in order to reduce the risk of human error.

2.2.11.2 Design for security properties changing over time

Security properties may change over time, for example when code is updated. SEMIAH needs to be designed with this in mind, so that the system easily can be adapted to handle new threats, vulnerabilities or design changes, for example replacing one security related component (e.g. authentication) with another.

2.2.11.3 Design with the ability to isolate or toggle functionality

It should be possible to turn off or contain compromised parts of the system, or to turn on performance-affecting mitigations, should the need arise (e.g. during a Denial of Service attack or flashcrowd). Not every identified vulnerability can be readily mitigated within a safe time period, and mission critical systems like SEMIAH cannot be taken offline until their vulnerabilities are addressed.

2.2.11.4 Design for changes to objects intended to be kept secret

Secrets, such as encryption keys and passwords may get compromised. Keeping secrets safe is a hard problem, and one should be prepared to have secrets replaced at any time and at all levels of the system. This includes several aspects:

- A secure way for users to change their own passwords, including disallowing the change until all required old authentication factors have been successfully presented by the user.
- Consider having a password recovery mechanism for forgetful users, where the user can reset their password after verification via a parallel mechanism (email, SMS or similar). Avoid providing the password in cleartext.
- Provide a secure and efficient way to replace digital certificates, SSH keys and other keys or authentication material that systems use. These events must be logged, without compromising the secrets involved. The logging mechanism should be secure and forensically verifiable (for example using external log servers and checksums).
- Make sure that key changes do not affect data at rest, for example data on an encrypted file system or database. There must be a way to upgrade the encryption key without losing access to the data.

2.2.11.5 Design for changes in the security properties of components beyond your control

Cryptographic ciphers may need to be changed, because of weaknesses, flaws, technology improvements or security vulnerabilities. External components security properties or related characteristics may also change over time, for example if a library being used is no longer being actively maintained or when its licence changes, forcing users to abandon it. In these cases it is important to design for agility – the capability to change layers and algorithms as needed in the system. One good example is Java’s possibility to change crypto providers and Apache’s negotiation of acceptable ciphers. Good design allows for intermediate layers of abstraction between code and imported external APIs so that developers can change components providing necessary functionality without changing much of the code.

2.2.11.6 Design for changes to entitlements

SEMIAH must have a way to revoke access to users or roles when a user no longer has access to them. Verifying this revocation mechanism should be a necessary part of the auditing procedures that are part of SEMIAHs security management process, in order to confirm that changes in access rights are being propagated properly, as well as verifying that the access control procedures do not risk that the organisation shoots itself in the foot by removing all access to a necessary user or role, essentially losing control over parts or all of the system.

We should also avoid storing sensitive information in cleartext in the gateway.

2.2.12 Protection schemes and policies for SEMIAH (T8.2)

This covers built-in Security and Privacy, defence-in-depth, protection schemes, privacy policies and access control policies for SEMIAH components. See also security model (appendix).

2.3 Consideration of Privacy by Design

SEMIAH aims at implementing Privacy by Design, which means adhering to the seven foundational privacy by design principles [7]. The objectives of privacy by design is ensuring privacy and gaining personal control over one’s information. Privacy by design may also give a competitive advantage for SEMIAH by providing better management of users’ private or confidential information than other market actors.

The European Parliament already stated that Europe needs a uniform and strong data protection law. The fundamental principles and the architecture of a new data protection reform proposal got positive feedback from several Committee. According to the planned new EU privacy directive privacy by design will be mandatory. [16]

The seven foundational privacy by design principles are [7]:

1. Proactive, not reactive; Preventative not remedial. SEMIAH should aim at utilising proactive measures for protecting private or confidential information, where the design anticipates and prevents privacy-invasive events before they happen.
2. Privacy as the default setting. SEMIAH should aim at protecting users’ personal data by default. This means that if the user does nothing, then their privacy remains intact.
3. Privacy embedded into design. SEMIAH aims at embedding privacy into the design and architecture of the Demand Response system. It will not be bolted on as an add-on after the fact. This means that privacy becomes an essential component of the core functionality being delivered. Privacy will be integral to SEMIAH, without diminishing functionality.
4. Full functionality – positive sum, not zero sum. Privacy by design aims at accommodating all legitimate interests and objectives in a positive sum – using a “win-win” approach. This means that privacy is not being traded off against security, but rather that privacy and security functionalities complement and enhance each other. This is not only a theoretical statement. We expect that there for example will be direct synergy between functionality required to protect confidentiality and privacy, for example in the case of secure logging systems providing forensics capabilities of who did what for sensitive system operations. The same base technologies can be used for protecting the users’ private data [8]. This is also apparent by comparing the previous section on security by design, which has several partly overlapping objectives with privacy by design, for example on how to ensure data confidentiality, strong authentication and keeping the user at the centre of focus.
5. End-to-end security – full lifecycle protection. Privacy by design should apply from private or confidential data is being created and until it can be securely destroyed in a timely fashion. SEMIAH may need to utilise several cryptographic techniques in order to achieve this, including end-to-end link encryption, encryption of sensitive data at rest as well as

methods for secure destroying of such sensitive data, for example by invalidating keys. This ensures that also all backup copies of such data are being invalidated. It furthermore means that strong, centrally managed access control mechanisms should be used, in order to ensure that sensitive data can be destroyed at end of subscription.

6. Visibility and transparency – keep It open. This ensures that all stakeholders can be assured that SEMIAH is operating according to the stated promises and objectives, subject to independent verification. Its components and operations remain visible and transparent to users and providers.
7. Respect for user privacy – keep it user-centric. This means that SEMIAH should keep the interests of the individual uppermost by offering such measures as strong privacy defaults, appropriate notice and empowering user-friendly options.

3 Methods and Tools for Security and Privacy Management for SEMIAH -

3.1 Risk assessment/gap analysis, safeguards, vulnerability management

In ISO31031, risk assessment is said to consist of:

- Risk identification – finding, recognising and recording risks.
- Risk analysis – understanding the risk.
- Risk evaluation – determine the significance of the level and type of risk.

Risk assessment for SEMIAH will primarily be based on MAGERIT version 3.0, as described in the following. This is a freely available standard which has been adapted to use in critical infrastructures in the PRECYSE FP7 project.

3.1.1 MAGERIT-based Risk Assessment

MAGERIT is a methodology developed by the Spanish Council for Electronic Administration. It consists of two main parts; *risk analysis*, and *risk management*.

Risk analysis helps to identify risks before they can have a negative impact on the organisation. Main elements to be analysed are:

- Assets that give value to the organisation.
- Threats that can cause damage to the assets and therefore also to the organisation.
- Safeguards or countermeasures they can be deployed to mitigate the threats and minimise damage to assets.

Based on this, *impact* (severity of the risk or damage potential) and *risk* (expected annual impact) can be estimated.

Risk management concerns establishing a thorough defence in order to prepare for emergencies, survive incidents and keep the organisation operational.

3.1.1.1 Assets

Important assets to take into account include:

- Data
- Services
- Computer applications
- Computer equipment
- Data storage
- Auxiliary equipment
- Communication networks
- Facilities that house the equipment
- People that operate the elements above

It is important to note that assets are not independent entities but has dependencies that mean that if an asset is affected by a risk, its dependent assets as well as assets that depend on it may also be affected. To simplify the dependencies, groups of assets may be structured into layers where higher layers depend on the lower ones.

- Layer 1: The environment. including equipment, power, communication lines, personnel, buildings.
- Layer 2: The information system. Including hardware, applications, communications, information media.
- Layer 3: Information. Data and meta-data.
- Layer 4: The essential functions of the organisation. Objectives, goods and services produced.
- Layer 5: Other assets: credibility, knowledge, privacy and confidentiality.

Important dimensions to an asset are: availability, authenticity, confidentiality, integrity, accountability. What damage would be caused if these attributes are affected by a threat?

Based on an understanding of an asset and the importance of its dimensions, it is possible to estimate its *value* by determining the total cost of an incident that destroys the asset. Note that assets lower in a dependency tree can accumulate the value of assets that depends on them. Valuation can be done qualitatively or quantitatively, and although requiring a great effort, a quantitative approach has many advantages. If the value is expressed monetarily, it is possible to analyse e.g. investments in safeguards and insurance.

3.1.1.2 Threats

Threats may come in many different forms, ranging from natural disasters, industrial accidents, or intentional attacks.

Threats that may affect and cause damage to each type of asset are to be identified and valuated. It is important to determine which dimensions of an asset are affected, and in particular the valuation depends on two factors: the degradation to the asset, and the frequency of the threat.

3.1.1.3 Impact

Accumulated impact is calculated for an asset taking into account the accumulated value of the asset, and the threats that it is exposed to and degradation caused, for each evaluation dimension.

3.1.1.4 Risk

Risk is a measurement of probable damage to the system, given the impact of the threats to the assets. Accumulated risk is calculated by taking into account the accumulated impact on an asset from a threat, and the frequency of the threats. The accumulated risk is calculated for each asset, threat and dimension.

3.1.1.5 Safeguards and Countermeasures

The impacts and risks are calculated without safeguards and countermeasures in place, thereby calculating the impacts and risks to the unprotected assets. Safeguards and countermeasures are procedures and mechanisms that reduce the risk. They may be proactive (reducing the frequency of the threat, preventing it from damaging the asset) or retroactive, reducing the impact of the threat and degradation of the asset.

Appendix C presents the complete set of safeguards and countermeasures described by MAGERIT V3 Book II, in English translation from the original Spanish.

With perfect safeguards and countermeasures, there will be no more impact or risk. However, it is usually not possible to implement 100% effective safeguards and countermeasures. With safeguards and countermeasures in place, residual impact and residual risk can be calculated. Residual impact takes into account the reduced degradation based on safeguards in place. Residual risk is based on residual impact and the reduced frequency of occurrence.

3.1.1.6 Risk Management

When impacts and risks are calculated, risk management include analysing the impacts and risks and the identification of appropriate safeguards and countermeasures as well as setting up appropriate policies, standards and procedures to ensure that each threat has a suitable response.

3.2 Tests and Metrics for assessing Security & Privacy

In addition to risk analysis, there are several tests and metrics that are relevant for assessing the Security and Privacy of the SEMIAH system.

3.2.1 Automated tests

Automated tests can be expressed in the XML-based Open Vulnerability Assessment Language (OVAL) [9]. OVAL provides an approach for determining if a vulnerability, software application, patch or configuration exists in a given system, and reporting the results of an assessment. There exist libraries of OVAL tests for different platforms that can be used as a basis for selecting an appropriate set for a given system. We can when necessary create customised OVAL tests for checking vulnerabilities of our own software components in SEMIAH.

3.2.2 Manual tests

The XML-based Open Checklist Interactive Language (OCIL) [10] can express compliance checks that require manual user interaction. It has tool support for presenting the questionnaire to the user, receiving the responses and reporting the results. A questionnaire of relevant compliance tests will be developed for SEMIAH and used as part of the security and privacy evaluation for the system.

3.2.3 Examples of relevant Metrics

Important metrics for evaluating resilience include measures of availability and reliability. Mean Time to Failure (MTTF) and Mean Time to Repair (MTTR) are obvious candidates.

The Entropy-based Privacy Leakage Metric estimates leakage of sensitive information in IDS alarms [11].

3.3 Anonymisation, pseudonymisation, encryption to reduce identified privacy leakages

The Reversible anonymiser for XML documents is a tool that supports anonymisation of XML documents and messages using an eXtensible Access Control Markup Language (XACML) anonymisation and authorisation policy [12]. This is is one of the core technologies that will be used for enhancing privacy, enforcing confidentiality and supporting access control in SEMIAH. The reversible anonymiser is available at http://launchpad.net/reversible.

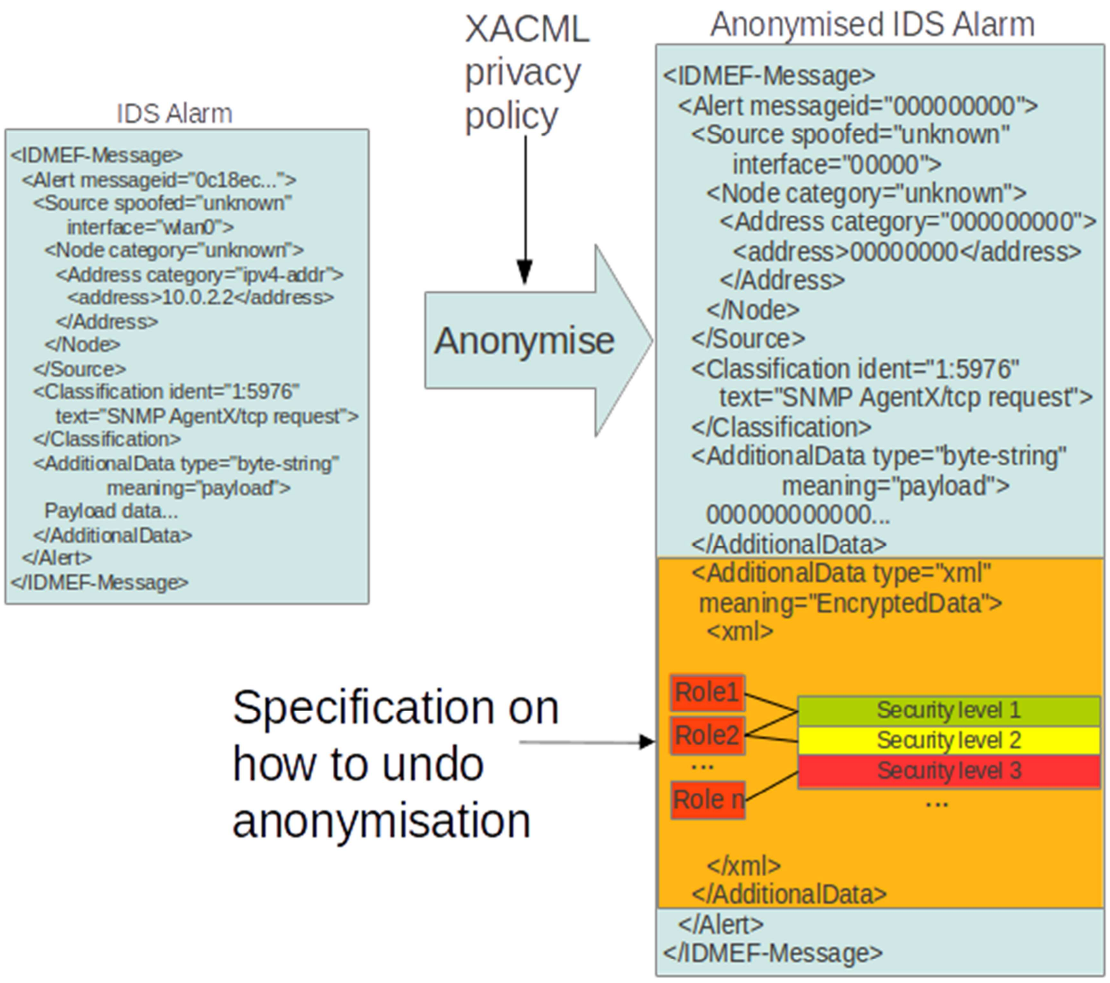
The reversible anonymiser supports transport layer security using HTTPS and the XACML policies ensure that only authorised users or services can reverse the anonymisation for private or confidential information. It furthermore supports multilevel security, meaning that information at different security levels or belonging to different stakeholders can be protected by encryption, so that only authorised stakeholders can access this information. The anonymiser furthermore supports location-based anonymisation or authorisation policies based on the GeoXACML framework [13]. The anonymiser currently supports anonymising Intrusion Detection System (IDS) alarms in the Intrusion Detection Message Exchange Format, which is an XML format for IDS alarms defined by IETF [14].

Future work that will need to be done in order to use the anonymiser with SEMIAH is:

- Support anonymisation for other relevant XML protocols that SEMIAH needs. This is typically protocols for transporting private or confidential information.
- Add support for anonymising JSON messages may be necessary.

Support for anonymising JSON messages can be implemented by adding support for defining anonymisation rules in JsonPath instead of XPath which currently is being used. In addition, some minor adaptations are needed, to generalise the handling of where the specification on how to reverse anonymisation will be inserted.

The reversible anonymiser will be used for implementing privacy-enhanced intrusion detection services, in order to detect attacks on the SEMIAH infrastructure. It may also be used to anonymise private or confidential information from the OGEMA front-end being stored in the back-end cloud as well as for implementing other necessary services, for example secure logging.



```
<IDMEF-Message>
 <Alert messageid="0c18ec...">
  <Source spoofed="unknown"
      interface="wlan0">
   <Node category="unknown">
    <Address category="ipv4-addr">
     <address>10.0.2.2</address>
    </Address>
   </Node>
  </Source>
  <Classification ident="1:5976"
    text="SNMP AgentX/tcp request">
  </Classification>
  <AdditionalData type="byte-string"
          meaning="payload">
   Payload data...
  </AdditionalData>
 </Alert>
</IDMEF-Message>
```

```
<IDMEF-Message>
 <Alert messageid="000000000">
  <Source spoofed="unknown"
      interface="00000">
   <Node category="unknown">
    <Address category="000000000">
     <address>00000000</address>
    </Address>
   </Node>
  </Source>
  <Classification ident="1:5976"
    text="SNMP AgentX/tcp request">
  </Classification>
  <AdditionalData type="byte-string"
          meaning="payload">
   000000000000...
  </AdditionalData>
  <AdditionalData type="xml"
   meaning="EncryptedData">
   <xml>

   </xml>
  </AdditionalData>
 </Alert>
</IDMEF-Message>
```

Figure 1 Overview of reversible anonymisation of XML documents.

3.3.1 XACML policy editor

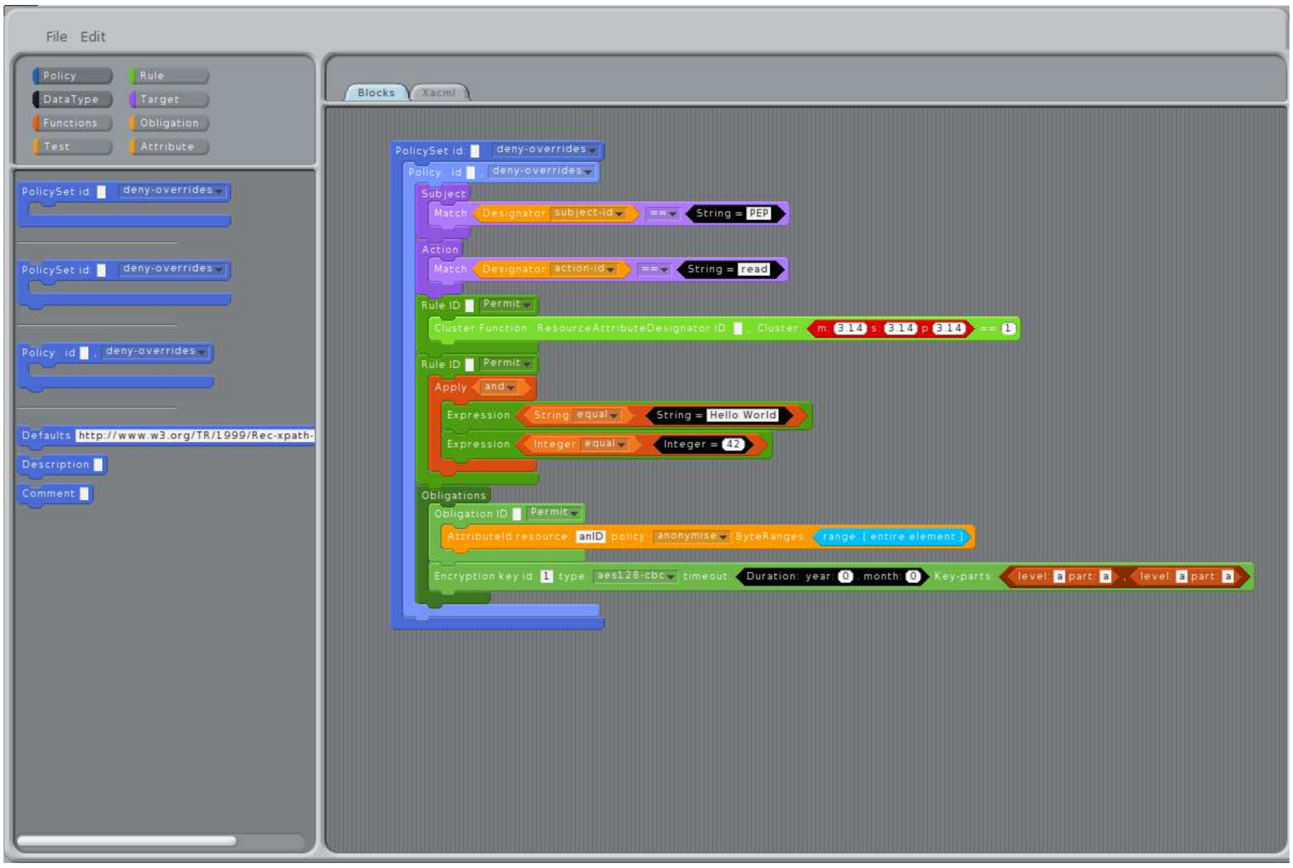

File Edit
Policy
Rule
DataType
Target
Functions
Obligation
Test
Attribute
Blocks
Xacml
PolicySet id deny-overrides
Policy id deny-overrides
Subject
Match Designator subject-id == String = PEP
Action
Match Designator action-id == String = read
Rule ID Permit
Cluster Function ResourceAttributeDesignator ID Cluster m 3.14 s 3.14 p 3.14 == 1
Rule ID Permit
Apply and
Expression String equal String = Hello World
Expression Integer equal Integer = 42
Obligations
Obligation ID Permit
AttributeId resource anID policy anonymise ByteRanges range [entire element]
Encryption key id 1 type aes128-cbc timeout Duration year 0 month 0 Key-parts level a part a level a part a
Defaults http://www.w3.org/TR/1999/Rec-xpath-
Description
Comment

Figure 2 XACML policy editor.

The XACML policy editor ViSPE is a related project which has defined an easy to use policy editor for XACML authorisation and anonymisation policies [15]. The policy editor is implemented using the Pharo Smalltalk engine and is based on the programming language Scratch (http://scratch.mit.edu). The policy editor provides graphical policy language that is easy to use for policymakers, and removes much of the complexity in managing the complex XML-based XACML policies. This policy editor will be used by SEMIAH to manage privacy and authorisation policies in XACML.

3.3.2 Example use case of Reversible Anonymiser

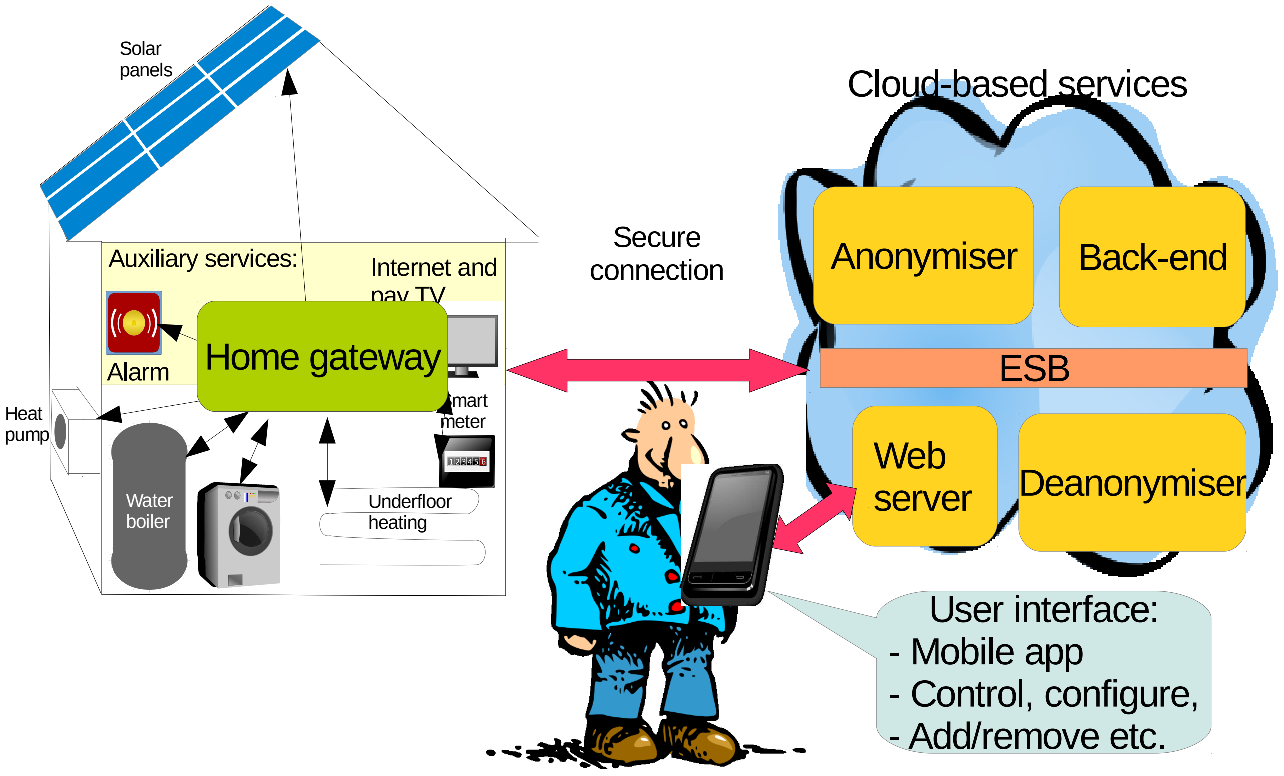


Figure 3 Example use case: anonymising private information from user.

Figure 3 shows an example use case of the reversible anonymiser. It is here assumed that the home gateway may produce some information that may be sensitive, for example usage patterns of electrical devices. This information may be both sensitive from a security perspective (usage patterns can for example be used for planning burglary of the home) and sensitive from a privacy perspective (giving private information about how electrical devices are being used). It is assumed that the home gateway can use secure services for sending this information to the cloud-based services, which are being controlled by an Enterprise Service Bus (ESB) or Message Oriented Middleware (MOM) which allows anonymisation and deanonymisation proxies to be connected when services that can contain private or confidential information are being used.

The anonymiser can then anonymise the given services, and store the sensitive information inside the messages, so that only authorised personnel later can undo this anonymisation by providing access to a private key that gives access to the given information sensitive information. This allows the user to get access to his/her own private information via the mobile user interface, for example information about service usage, billing information etc., at the same time as sensitive part of this information can be restricted for other parties. This means that service providers can get sufficiently detailed information to run the services, but not details at the level which violates privacy. The privacy policies may furthermore support that system administrators or others can investigate the detailed information, for example for debugging or troubleshooting, but such access to private or confidential information would be required to be logged, in order to provide transparency on access to private or confidential information. The log server would need to support secure logging, so that the log data cannot easily be manipulated or tampered with by the service provider, in order to provide transparency on access to sensitive data.

The mobile user interface would in a similar way support privacy enhanced operation, so that sensitive information, for example user preferences, could be protected so that only the user and services which need the given information can access it.

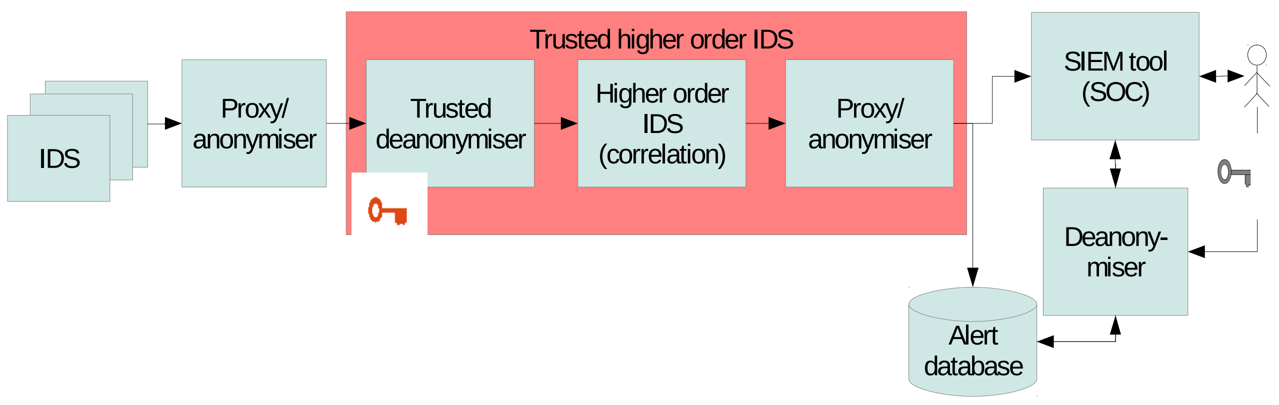
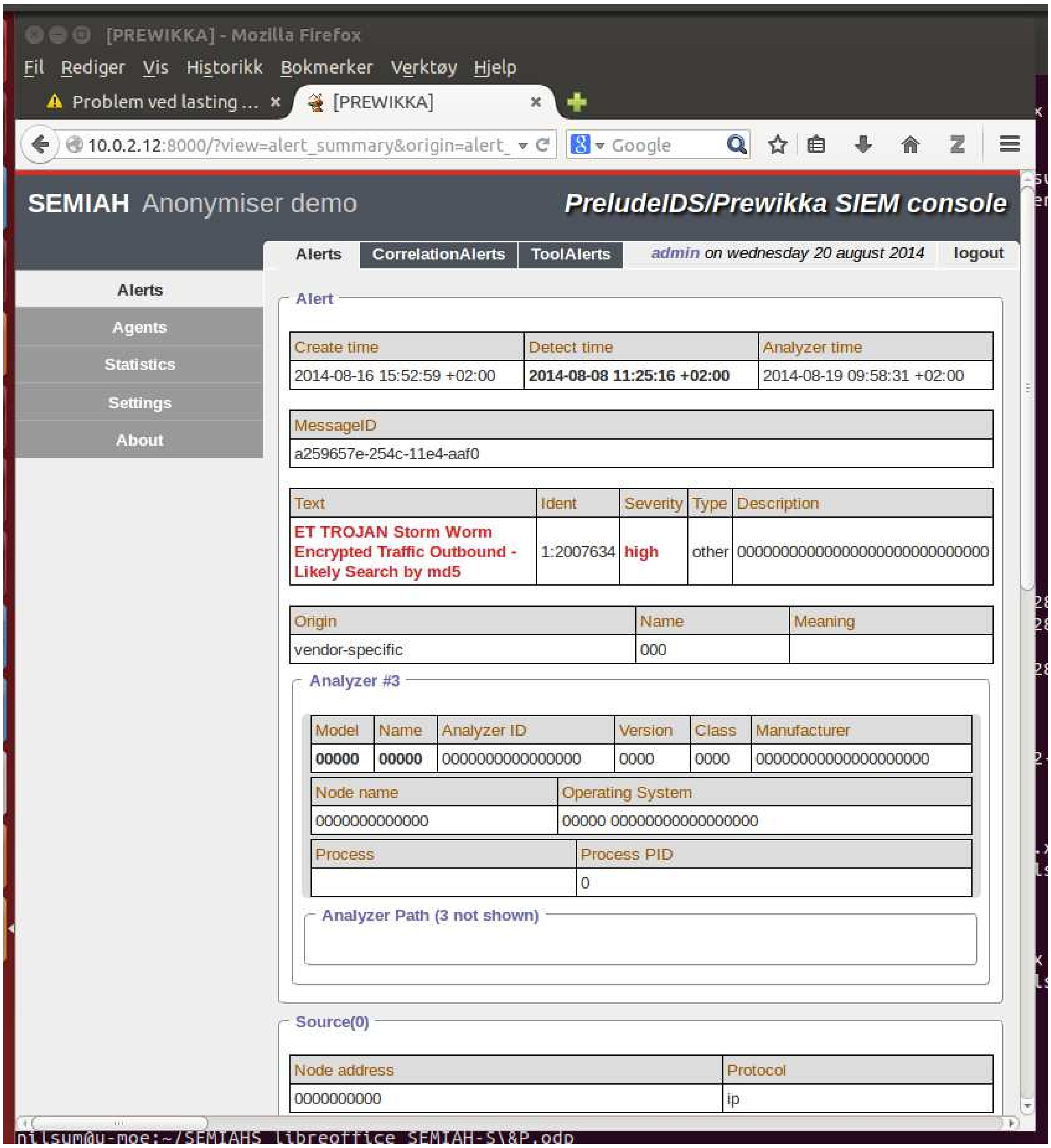


Figure 4 Privacy-enhanced IDS scenario.

Figure *4* illustrates how privacy-enhanced Security Information and Event Management systems (SIEM) can be implemented. IDS alarms from a set of IDS sensors can when necessary be sent through the anonymiser proxy, which anonymises the alarms according to an XACML privacy policy. A privacy impact assessment can be used for defining the privacy policy deciding which alarms from which IDS sensors that need to be anonymised. The anonymised alarms can if necessary be sent through an alarm correlation system, which may be trusted to do data mining of some of the anonymised parameters. The alarm correlation system can do this using its secret key. Alarms triggered by the alarm correlation system can subsequently be anonymised when necessary, and stored anonymised in an alert database, as well as presented to the security operator via a Security Incident and Event Mangement (SIEM) system running in the Security Operations Centre (SOC). The security analyst or other authorised stakeholders (e.g. a CERT team) may use their respective private keys for on-demand deanonymisation of information they are authorised to see in the IDS alarms. Figure 5 shows how IDS alarms in IDMEF format appear in the SIEM tool (Prelude-IDS), after being anonymised by the reversible anonymiser. Future work that needs to be implemented is adding support for on-demand deanonymisation of IDS alarms together with secure logging of such events in order to provide transparency on access to sensitive information. This can for example be implemented by storing the IDS alarms in an XML database and adding a query interface for retrieving and deanonymising these alarms and subsequently presenting them in the SIEM interface. There are also some other improvements that need to be implemented, for example the subscribe part of the publish/subscribe interface, so that the SIEM can subscribe to IDS alarms.

3.3.3 Example use case: Privacy-enhanced intrusion detection systems.


[PREWIKKA] - Mozilla Firefox
Fil Rediger Vis Historikk Bokmerker Verktøy Hjelp
Problem ved lasting ...
[PREWIKKA]
10.0.2.12:8000/?view=alert_summary&origin=alert_
Google
SEMIAH Anonymiser demo
PreludeIDS/Prewikka SIEM console
Alerts
CorrelationAlerts
ToolAlerts
admin on wednesday 20 august 2014
logout
Alerts
Agents
Statistics
Settings
About
Alert
Analyzer #3
Analyzer Path (3 not shown)
Source(0)
nilsum@u-moe:~/SEMIAHS libreoffice SEMIAH-S\&P.odp

Create time	Detect time	Analyzer time
2014-08-16 15:52:59 +02:00	**2014-08-08 11:25:16 +02:00**	2014-08-19 09:58:31 +02:00

MessageID
a259657e-254c-11e4-aaf0

Text	Ident	Severity	Type	Description
ET TROJAN Storm Worm Encrypted Traffic Outbound - Likely Search by md5	1:2007634	**high**	other	00000000000000000000000000000

Origin	Name	Meaning
vendor-specific	000	

Model	Name	Analyzer ID	Version	Class	Manufacturer
00000	**00000**	0000000000000000	0000	0000	00000000000000000000

Node name	Operating System
0000000000000	00000 0000000000000000

Process	Process PID
	0

Node address	Protocol
0000000000	ip

Figure 5 Privacy-enhanced intrusion detection system.

3.3.4 Other possible use cases for the Reversible Anonymiser in SEMIAH

The Reversible Anonymiser may also be useful for other purposes in SEMIAH. One example is implementing secure configuration and policy management. Such functionality could utilise key shares in the anonymiser to implement separation of duties constraints where several stakeholders must agree to deploy a critical system configuration.

Another example is implementing a trustworthy policy administration point for XACML policies where a service would be able to deploy anonymised privacy and security policies, so that only authorised stakeholders or services can deanonymise and instantiate policies they have access to. This approach would keep sensitive parts of the authorisation policies secret, tamper-resistant and unreadable by any non-authorised stakeholder, at the same time as it would allow for parallelising the authorisation process by being able to run a distributed set of XACML authorisation engines.

It is also possible to cryptographically bind a system configuration to a workflow, so that only tested and approved system configurations can be deployed on the running system. This reduces the risk that faulty or malicious system configurations destabilises the Demand/Response system.

Another use case is to use the Reversible Anonymiser as a building block for implementing secure logging services. This can if necessary be combined with time-based data expiry, so that stored cryptographic data after a given time period becomes worthless by invalidating the encryption key [8]. A secure logging service is useful for SEMIAH for several reasons:

- Protect private or confidential information in the system logs;
- Support non-repudiation of actions and transactions in the system log. This means that a stakeholder cannot deny having accessed certain private or confidential information, since the system logs provides transparency on who did what;
- Provide forensics-grade logging services that cannot easily be manipulated. An insider wanting to manipulate the logs will not be able to get access and will not know which part of the log to modify.

The Reversible Anonymiser can also be used as a building block to implement a privacy-enhanced XML database.

3.4 Mapping ENISA proposed security measures for smart grids to SEMIAH

The ENISA smart grid task force has delivered a proposal for a list of security measures for smart grids [16]. This document provides European smart grid asset owners with a catalogue of 45 available security measures grouped in 11 domains, which might help them in improving the level of cyber-security in their installations. Data privacy is considered out of scope of this document, although the document stresses that smart grids should both operate securely and respect users' privacy.

The objectives of this activity are:

- Aligning the varying levels of security and resilience of the asset owners with a consistent minimum framework;
- Providing an indication of a minimum level of security and resilience in the Member States with regards to the smart grids, thereby avoiding the creation of the "weakest link";
- Ensuring a minimum level of harmonisation on security and resilience requirements for smart grids across Member States, and thus reducing compliance and operational costs; and
- Setting the basis for a minimum auditable framework of controls across Europe.

The asset owners should then perform a risk assessment in order to define appropriate security measures from this catalogue. A risk assessment should be performed throughout the life cycle of a smart grid infrastructure, in particular during requirements definition, procurement, control definition and configuration, system operations and end of system lifetime.

After the risk assessment, the asset owner should take reasoned decisions on which elements of the catalogue that are appropriate. This will depend on the impact on assets from materialised threats, their vulnerabilities and the threat exposure. The document furthermore maps between security measures (security controls), assets and threats.

The WP8 team has gone through the security controls in this document and proposed how SEMIAH can implement the proposed general security measures. This does not include the elements of secure software design, which has been covered earlier in this document.

3.4.1 Taxonomy of threats

ENISA has proposed a taxonomy of threats for smart grid services, which includes:

- Natural disaster (fire, flood, thunder, environmental disaster, etc.).
- Damage, loss of IT assets (damage by 3rd party, test corruption, loss of information integrity, loss or destruction of devices, media, documents, media, information leakage)
- Outages (loss of Internet, network, support services, energy, lack of resources, personnel, strike).
- Nefarious activity, abuse/cyber-attacks (ID theft, spam, DoS, malicious code/activity, social engineering, abuse information leakage, rogue certificates, HW/SW manipulation, manipulate information, misuse of audit tools, falsification of records, misuse of information, information systems, unauthorised: access, administration, software installation, software use, compromising confidential information, abuse authorisations hoax, badware, remote activity, targeted attacks).
- Deliberate physical attacks (bomb attack/threat, sabotage, vandalism, theft, information leakage, sharing, and unauthorised physical access).
- Unintentional damage (Erroneous: information sharing/leakage, use or administration of systems/devices, use of unreliable information, unintentional alteration of data, inadequate design, planning/adaptation).
- Failures/malfunction (Device/system failures, disruption of communication links, power supply failure, service provider failure, malfunction).
- Eavesdropping, interception, hijacking (wardriving, intercepting information, man in the middle session hijacking, repudiation of actions, reconnaissance/information gathering, replaying messages).
- Legal (Unauthorised use of copyrighted material, failure to meet contractual obligations, violations of laws).

3.4.2 Risk handling

Risks should be detected, and when detected, one of three risk mitigation strategies must be chosen:

- Risk reduction (preventative)
- Risk mitigation (after the fact) or
- Risk acceptance

Risk detection, in order to identify the risk can be done using sensors and tools like the OpenVAS vulnerability scanner, but also network management systems (e.g. OpenNMS or Nagios) as well as using intrusion detection systems.

Risk reduction (preventative) means reducing the impact and/or expected frequency of risks before they occur. Methods for risk reduction include adding safeguards or countermeasures against the risk, such as: redundancy, ensuring geographical diversity, active response against attacks/intrusion prevention, contingency planning, emergency plan, exercises, insurance, condition-based maintenance, testing/fuzzing to detect software flaws, software code inspection etc.

Risk mitigation (after the fact) includes restore the system using a working backup or virtual machine snapshots, perform attack containment for example by reconfiguring firewalls to isolate the attacked system or taking affected systems offline for forensics investigation.

Risk acceptance means that if the risk is considered tolerable, then one might decide to live with the risk without doing any risk mitigation.

3.4.3 Appropriate security measures

ENISA proposes security a set of security domains, which are appropriate security measures for smart grid installations. The next subsections go though each security domain and proposes how SEMIAH can implement technical security controls for these.

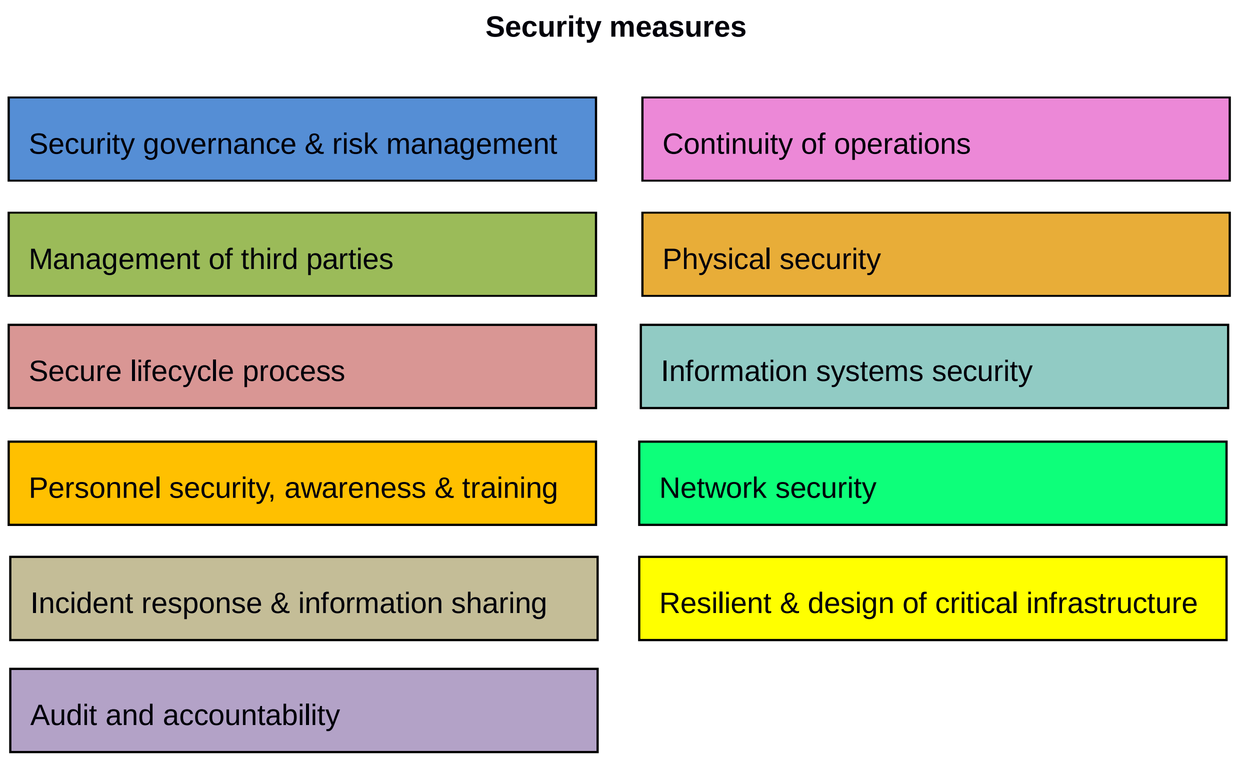

Security measures
Security governance & risk management
Continuity of operations
Management of third parties
Physical security
Secure lifecycle process
Information systems security
Personnel security, awareness & training
Network security
Incident response & information sharing
Resilient & design of critical infrastructure
Audit and accountability

Figure 6 Security measure domains.

3.4.3.1 Domain 1: Security governance & risk management

Security governance and risk management consists of the following elements:

Information security policy

This involves writing and maintaining a high-level written policy

Organisation of information security

This includes managing security roles and responsibilities. Technical enforcement can be authentication mechanisms (e.g. the Security Assertion Markup Language (SAML) or OpenID) and authorisation mechanisms (e.g. XACML). The Reversible Anonymiser might be a useful component here [8], as a policy enforcement point.

Information security procedures

These are procedures (manual or automated) that support the information security policy. For example: XACML privacy policies or authorisation policies, firewall rules, IDS rules, configuration management system configuration, written procedures etc.

Risk management framework

The risk management framework manages assets, vulnerabilities, threats and threat frequency and risk. It will be integrates with vulnerability assessment systems and asset scanning systems. Aggregated risks must be supported, for example using an attack tree or attack graph based approach.

It must support existing security management standards, for example the ISO27000 set of security management standards, the German BSI Grundschütz or the open Spanish standard MAGERIT. Verinice (http://www.verinice.org) is an open source risk assessment framework that supports most of these features. The PRECYSE project has created a MAGERIT catalogue of countermeasures that could be used as a baseline. This catalogue is free as opposed to the ISO27000 and BSI Grundschütz catalogues for Verinice.

Verinice can be integrated with the open source vulnerability scanning tool OpenVAS (http://www.openvas.org), so that the results of vulnerability scans can be imported into the risk assessment tool. OpenVAS can furthermore be integrated with vulnerability tests written in the Open Vulnerability Assessment Language (OVAL). This allows for performing passive vulnerability tests on hosts instead of using active vulnerability tests, explicitly testing for vulnerabilities. The latter may be problematic to perform on a critical infrastructure in operation. However nonintrusive OVAL tests can also be performed on systems in operation.

Risk assessment should be performed at regular intervals

This includes performing vulnerability scanning, then reassessing identified risks and other risks using a risk management tool to detect the gap in security.

Risk treatment plan

The risk treatment plan explains how to mitigate the identified risk. It can either be written manually, or be generated automatically for commonly occurring risks. Automatic management can for example be done based on reports in the Common Remediation Enumeration (CRE) format. Risk mitigation techniques are also often conveyed in vulnerability reports from the Common Vulnerabilities and Exposures (CVE) database. The risk can be considered as mitigated when a suitable security control has been activated to counteract the risk, so that any residual risk is considered acceptable.

3.4.3.2 Domain 2: Management of third parties

Third party agreements are needed

They should ensure that availability, integrity and confidentiality is preserved. The agreement should furthermore be incentive compatible, if possible, meaning that the third party does not benefit from cheating or doing its duties poorly.

Monitor compliance of contractual obligations

Using key performance indicators and clear acceptance criteria. Monitoring techniques can for example be auditing using interactive tests in the Open Checklist Initiative Language (OCIL). Technical means should be implemented in order to make the third party operation auditable, for example using secure logging services which guarantee non-repudiation, so that the third party cannot deny having done a certain operation. The secure logging scheme should also ensure transparency, so that SEMIAH systems can verify that the operation is being performed according to the contract. In other words, trust, but verify.

The Reversible Anonymiser (http://launchpad.net/reversible) could be a useful building block for implementing secure logging services.

3.4.3.3 Domain 3: Secure lifecycle process

Perform security requirements analysis and specification

The security requirements analysis ends up in a requirements specification document (e.g. D3.2 and this document).

Create an inventory of smart grid components/systems

The risk assessment system Verinice will be used for managing an inventory of assets. OpenVAS can be integrated with the passive realtime asset detection system (PRADS), in order to populate or perform consistency checks on the catalogue of IP assets in the risk assessment framework. OpenVAS can furthermore be integrated with Verinice.

Secure configuration management

This involves ensuring secure deployment of system configurations, for example using digitally signed system configurations. The integrity of deployed system configurations can if necessary be verified using tools supporting file integrity checks (for example OSSEC, http://www.ossec.net).

Multi party authorisation based on key shares can be used to implement separation of duties constraints, for example to ensure that more than one stakeholder must agree to deploy a given configuration, or to enforce that a standard workflow has been followed where configurations have been verified in testing before they are being deployed. Secure configuration management also includes the possibility to roll back to the last known good configuration.

RFC 6241 NetConf could be a possible candidate standard for handling configuration management in SEMIAH. The Reversible Anonymiser could also be useful for enforcing multi party authorisation and configuration deployments (in particular deployment of privacy/authorisation policies).

Secure documentation of configurations

The documentation of system configurations should be considered sensitive information that can be abused if it falls in the wrong hands. Such information should therefore be handled as confidential information, for example by enforcing access control on who can read this information, secure logging of who have read the information as well as secure archiving when the information is at rest.

The Reversible Anonymiser can be used as building block for implementing secure documentation of configurations.

Maintenance of smart-grid components

The configuration management system should enforce installation of only digitally signed software, firmware, patches and configurations.

Disposal of smart-grid components/systems

Ensure that procedures exist for wiping or destroying components that may contain sensitive information, to avoid information leakage when equipment is being decommissioned.

Change management

Ensure that only authorised and tested configuration changes can be deployed. This can for example be done by using a cryptographically enforced workflow for deploying configuration changes. There must be a possibility to roll back to the last known good configuration.

Security testing of smart grid components/systems

This can be used using vulnerability scanning tools like OpenVAS and OVAL tests, as described earlier.

3.4.3.4 Personnel security, awareness and training

Personnel screening

The HR department needs to perform background checks and security vetting of personnel that may have access to private or confidential information and also personnel that are allowed to make any changes to the system. This must be reflected in the security levels that a user can access.

The Reversible Anonymiser can be used as a tool for enforcing access to information at different security levels.

Personnel changes

There must be routines and technical systems that can create, modify or revoke user accounts (only personal accounts are allowed).

SEMIAH can use of existing federative access control systems here, for example Shibboleth (http://shibboleth.net) or OpenID (http://openid.net). These are scalable federated identify solutions that amongst others supports authentication and authorisation of users.

Security training and certification

It is important to establish and maintain security awareness in the organisation. This means that a training programme, exercises etc. are needed.

3.4.3.5 Domain 5: Incident response & information knowledge sharing

It is important that the organisation has incident response capabilities

For example a 24x7 managed security service detecting attacks This includes having intrusion detection and prevention systems installed. Attack response procedures and digital forensics capabilities are also needed. SEMIAH can make use of several existing technologies for this, for example Snort (http://www.snort.org) or Suricata (http://www.suricata-ids.org) for network-based intrusion detection, Prelude-IDS (http://www.prelude-ids.org) or similar for implementing a Security Incident and Event Management System (SIEM). There will probably need to be a separate intrusion detection system for OGEMA in order to detect attacks and abuse of the home energy management gateway. This will probably be a new software module, unless it can be installed as an operating system service below OGEMA. This can also be used for detecting system faults.

Routines are required in order to quickly fix a problem and restore services to normal operation. Efficient backup routines and/or virtual machine snapshots can be used to restore core services quickly after an attack.

Private or confidential information can be protected by utilising the Reversible Anonymiser. There will probably also need to be a centralised alarm correlation system, in order to reason on and reduce the total number of alarms. Prelude-IDS is one tool that can be used as a building block for this.

Vulnerability assessments

Software vulnerabilities in existing software modules can be detected using vulnerability scanning tools like OpenVAS together with OVAL tests. Vulnerabilities in own software can be detected using software code inspections and fuzzing test tools (e.g. fuzzdb) that tries to impose random, but plausible input within given bounds to a given software module, in order to detect crashes or abnormalities due to poor input validation.

Vulnerability treatment

This includes configuration management and patching routines, as has been discussed earlier. Information on how to detect and mitigate vulnerabilities should be shared with peer organisations and security organisations if possible:

- For example by registering CVE vulnerabilities in the Mitre database;
- Informing national Computer Emergency Response Teams about attacks;
- Exchanging threat information with peer organisations using the Structured Threat Information eXchange (STIX) and Trusted Automated Exchange of Indicator Information protocols may also be considered.

3.4.3.6 Domain 6: Audit and accountability

SEMIAH should include the following mechanisms in order to ensure auditability and accountability:

Auditing capabilities

A central component for ensuring auditability and accountability is using a secure logging scheme. This means using techniques like remote logging as well as storing the logs in a tamper resistant manner, so that only authorised stakeholders can access log information on a needs basis. The SEMIAH tools will need to implement logging operations at appropriate points in the source code, in order to log when sensitive operations are being performed. These points will need to be identified during system design.

Monitoring of smart grid information systems

In addition to logging critical operations in own source code, several other tools and techniques can be used by SEMIAH. For example:

- Intrusion detection systems for logging attacks and security policy violations;
- Network monitoring (e.g. OpenNMS) for monitoring service operation;
- Integrity checking tools for monitoring file access (e.g. OSSEC).

Protection of audit information

The secure logging system should also support nonrepudiation, so that access to sensitive commands or information cannot be denied, in order to provide transparency. It may need to implement a time-based encryption scheme, where data automatically is invalidated after a given time period, for example to support data retention mechanisms, or to comply to data protection requirements. This ensures that sensitive material in backup copies cannot be accessed after the legal data retention period has expired. This can be done using key expiry time cryptographically bound to the encryption key. The time expiring encryption key is derived from a master key using key derivation function [17], [18].

Encryption is a key technology for keeping confidential audit log information secure. The Reversible Anonymiser can be used as a building block for implementing secure logging services and similar services, like privacy-enhanced IDS operation.

3.4.3.7 Domain 7: Continuity of operations

Continuity of operations entails the following:

Maintain essential functions after disruption

Detect disruption using network management tools (e.g. OpenNMS), alternatively using intrusion detection systems or information provided by the users. Then restore operation according to defined and tested procedures as fast as possible, for example using backup media or virtual machine snapshots for cloud-based services.

If operation is being disrupted due to a Denial of Service attack or lack of resources, then one way to mitigate the attack might be to scale up the service, assuming that critical parts are implemented using a cloud provider platform that allows for elastic scalability according to need (i.e. using Platform as a Service).

Compliance tests can be used to verify that the service continuity procedures work as expected. OCIL tests can be used to implement such procedures. Guidance is needed to be able to understand what have happened, how to prevent it in the future and how to restore

Emergency communication services

Verify that emergency procedures and communication services work. OCIL tests can be used to check compliance for this.

3.4.3.8 Domain 8: Physical security

Important factors of physical security are:

Maintain physical security

OCIL compliance tests can be used for verifying procedures.

Log and monitor physical access

Physical alarms could be configured to send alarms to the IDS, so that the 24x7 operations centre could detect and handle such alarms. Privacy-enhanced operation should be preferred, for example based on the Reversible Anonymiser.

Physical protection of remote equipment

For example trigger an IDS alarm if smart grid equipment is being opened or tampered with.

3.4.3.9 Domain 9: Information systems security

Information systems security is a broad area that amongst others covers:

Policy for classification/disclosure of sensitive/secret information

The XACML privacy policy for the Reversible Anonymiser can be used for this.

Data security

Data at rest can be protected using disk encryption, database encryption or the Reversible Anonymiser. Data in transit can be protected using HTTPS/TLS or a VPN solution. Data integrity can be protected by using digital signatures, or checksums for less critical data. Tools like OSSEC can be used to verify file integrity. Data availability can be ensured using backup or redundant storage media, database servers etc.

Account management

Single sign-on and directory services/LDAP can be implemented using for example Shibboleth or OpenID. Logical access control should support two factor authentication and XACML authorisation.

Secure remote access

Using virtual private networks (VPN) e.g. IPsec.

Information security

- SEMIAH back-end:
  - Cloud-based systems can run heavy information security tools, for example IDS tools (Suricata/Snort), SIEM tools (e.g. Prelude-IDS), host-based IDS (e.g. OSSEC), anti-virus (e.g. ClamAV), anti-spam (e.g. SpamAssassin).
- SEMIAH front-end:
  - The OGEMA front-end also needs protection, but less heavy solutions are needed, due to limited resources. For example a tailor-made host-based IDS solution based on log analysis. A network based light weight IDS with regex matching capability would also be useful
  - Firewall functionality can if needed also be built in to the front-end, since the underlying Linux operating system supports this.
    -
- Media handling;
  - Secure procedures are needed for access, storage, distribution and destruction of storage media. Sensitive data must be protected during the entire lifecycle.
  - This can be done using techniques like the Reversible Anonymiser, encryption or similar.

3.4.3.10 Domain 10: Network security

Requirements for achieving network security:

Functional and secure network segregation

The Domain/Enclave model can be used for segregating the network, where the Domain protects the network and the Enclave is the network being protected. This is the model used for network segregation in the PRECYSE project. Concerted firewall rules define the allowed traffic within an enclave, and denying everything else by default. IDS whitelist which monitors allowed traffic may be considered. Traffic that is not allowed will then cause IDS alarms. This should only be implemented in the case where the allowed information that crosses the IDS is very well defined, otherwise this may affect manageability.

Secure network communications

Encrypted links (e.g. HTTPS/TLS/IPsec) should be used where needed. Trust between services can be achieved by using digital certificates and requiring authentication by these certificates during link setup.

3.4.3.11 Domain 11: Resilient and robust design of critical infrastructure

Minimum threat exposure

This may be achieved using IDS whitelisting considering the possible solutions, restrictive firewall rules only allowing traffic that is explicitly permitted, vulnerability scanning tools (OpenVAS and OVAL) to detect known vulnerabilities, patching and upgrading with subsequent testing to remove vulnerabilities.

Resiliency

Resiliency can be achieved using tested crisis procedures, redundancy, graceful degradation during fault or attack and quick recovery after a crisis.

Safe continuity after interruption of services

To ensure safe continuity after interruption of services, several issues have to be considered. The geographical location of components should be considered, ensuring that “all eggs are not in the same basket”. Ensure that the service is resumed in a sensible state after interruption, or if it needs to be restored from a virtual machine snapshot or backup. Also ensure graceful degradation, for example so that the service still can be managed and kept working if part of the grid or communication networks fall out. Avoid that the critical parts of the infrastructure has hard dependencies to (i.e., it cannot function without) less critical parts.

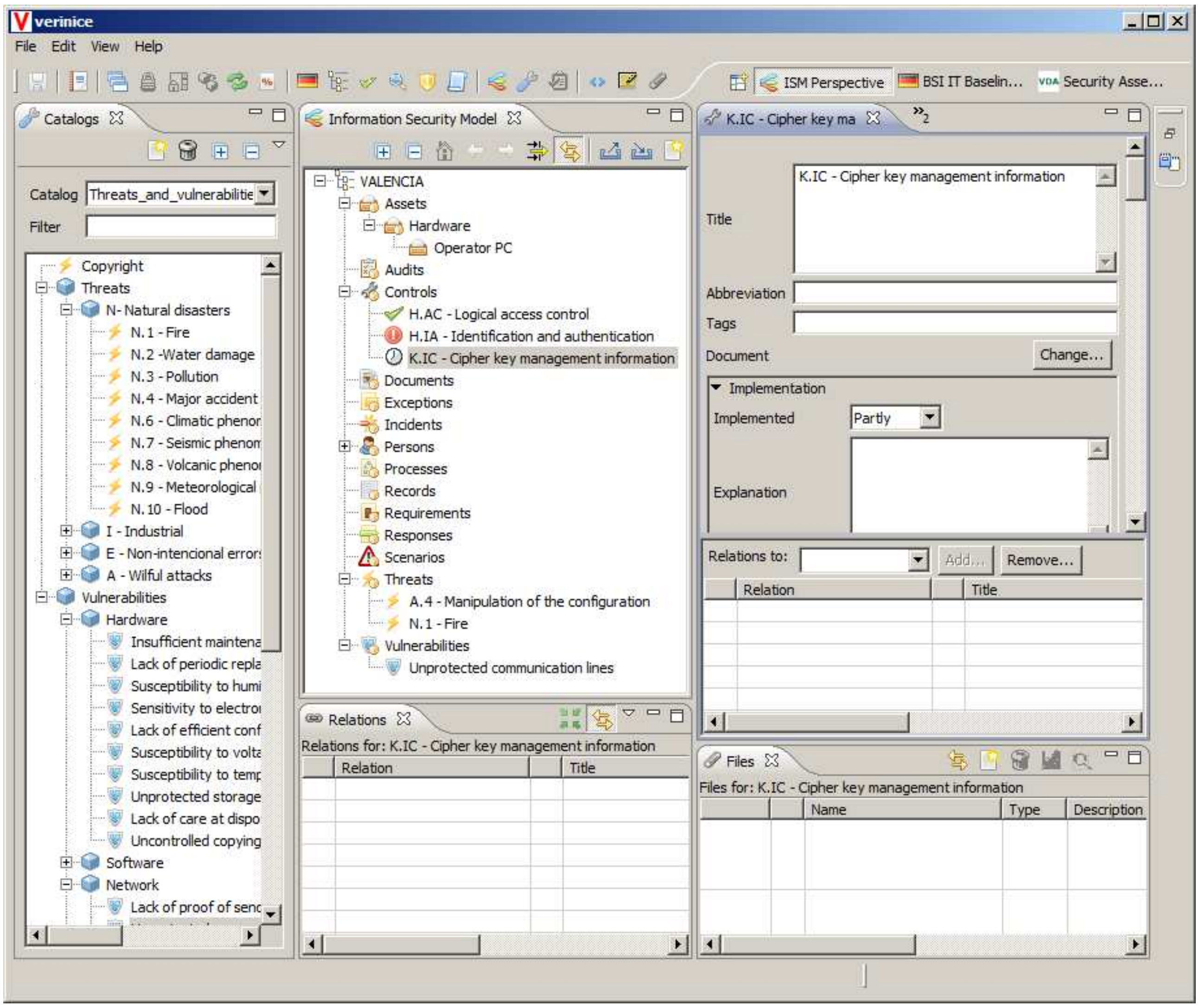
3.4.3.12 Survey of security control mapping

Based on the smart-grid security measures defined in the ENISA proposed security measures for smart grids, we have proposed an initial mapping from security domains to technical security controls, as shown in the table below.

	Security domain	**Technical security controls**
1	Security governance & risk management	Reversible, Verinice, OpenVAS, CRE
2	Management of third parties	Reversible, OCIL
3	Secure lifecycle process	Reversible, IDS/SIEM, OCIL, OVAL
4	Personnel security, awareness and training	Reversible, Shibboleth/OpenID
5	Incident response and information exchange	Reversible, IDS/SIEM, OpenVAS, OVAL, fuzzdb, CVE
6	Audit and accountability	Reversible, IDS/SIEM, OpenNMS, crypto
7	Continuation of operation	Reversible, IDS/SIEM, OpenNMS, OCIL, cloud services
8	Physical security	Reversible, IDS/SIEM, OCIL
9	Information security	Reversible, IDS/SIEM, OpenVAS, Shibboleth/OpenID, OVAL, crypto, VPN, Anti-virus, spamfilter, PRECYSE tools
10	Network security	Reversible, IDS/SIEM, crypto, firewall, PRECYSE

		tools
11	Resilient and robust	Reversible, IDS/SIEM, OpenVAS, firewall, PRECYSE tools, OVAL, cloud services

3.4.4 Important elements for ensuring information security


Catalog: Threats_and_vulnerabilitie

Title: K.IC - Cipher key management information

Implemented: Partly

Relations for: K.IC - Cipher key management information

Files for: K.IC - Cipher key management information

Figure 7 Verinice risk assessment tool.

SEMIAH needs a risk assessment tool and methodology. This need is also described in section 4 of D6.1, where both product and project risks must be considered in a risk management plan.

Our plan is to use the open source risk assessment tool Verinice (http://www.verinice.org/). This is a risk assessment tool that supports an attack tree based method for evaluating an overall aggregated risk for a given asset. It supports several security management standards, amongst others the ISO27000 set of standards, the German BSI Grundschütz standard. Support for both of these standards must be purchased, however the tool also supports the freely available Spanish standard MAGERIT. This catalogue of countermeasures is available as open source from the PRECYSE project. SEMIAH can use and adapt this catalogue according to the ENISA security measures for smart grid. Verinice can be used both for handling product product and project risks. Security and privacy risks will be a subset of these general risks. The risk management plan should include the risks on low voltage stability that are identified in Table 3 of D7.1, so that these risks are mapped down to regression test cases on grid constraints and similar, ensuring that the SEMIAH scheduler does not aggrevate stability problems in the grid, for example by shifting load too quickly..

Verinice supports an asset inventory, which allows for estimating the expected impact of materialised threats in order to calculate the risk. It also supports keeping record of identified vulnerabilities of assets, and can import vulnerability scan reports from the open source vulnerability testing tool OpenVAS (http://www.openvas.org). This allows for integrating vulnerability testing of SEMIAH with the risk analysis framework.

OpenVAS can also be used for identifying Internet Protocol (IP) based assets, by interfacing it with other tools, such as the passive realtime asset detection system (PRADS). This allows for providing an initial population of the asset directory in Verinice, if necessary, and it allows for consistency checking, in order to identify new assets, assets or services that should not be enabled etc.

3.4.5 Controlling privacy, confidentiality and information leakage

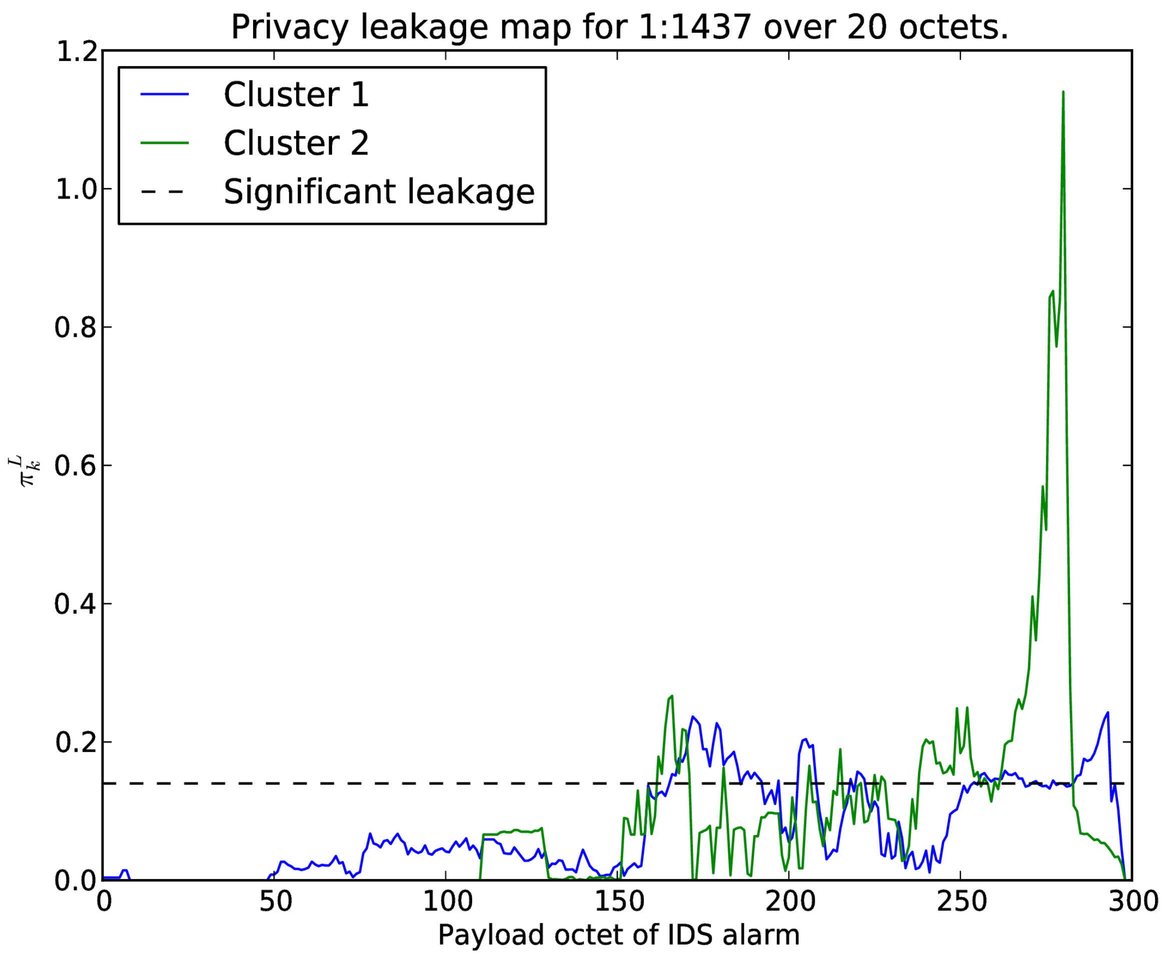


Figure 8 Example information leakage analysis of IDS alarm using privacy leakage calculator.

Data privacy controls are not being covered by the security controls proposed in the ENISA security measures for smart grids [16]. SEMIAH will therefore use the PhD work of Nils Ulltveit-

Moe as a methodology for controlling privacy leakage [19]. This methodology uses entropy-based privacy leakage risk metrics for:

- Information leakage detection;
- Privacy policy verification;
- Privacy risk gap analysis.

An example of a privacy leakage analysis for a Snort IDS rule (SID 1:1437 Windows Multimedia Download) is shown in Figure 7. The privacy leakage calculator initially performs a clustering into different attack vectors, and then plots the privacy leakage as the entropy standard deviation for each identified attack vector. This approach is tailored for detecting leakage in intrusion detection systems, but will also work for general anonymisation policy compliance verification. The Reversible Anonymiser is used as the main privacy enforcement mechanism in this methodology. The privacy leakage metrics can then be used to verify that a privacy policy is working as expected.

These entropy-based metrics may also be used in SEMIAH for detecting certain types of attacks, most notably Denial of Service attacks and perhaps also for detecting pre-boot attacks on the home energy management gateway and cloud-based servers. This shows that the privacy leakage metrics also may be useful from a security perspective, illustrating the “win-win” approach that Privacy by Design should aim at achieving.

3.4.6 OGEMA security and privacy

The OGEMA framework allows assigning permissions to users accessing the system via the graphical user interface, applications running on the gateway and to users that send external requests to the framework’s own REST interface. These permissions can be pre-configured or assigned/modified during runtime via the administrator web interface.

Access to applications’ web interfaces is performed via a browser and is https-encrypted. The login is performed with username/password. After a successful login, a user can access the user pages of those applications that he has been granted access to. When a user can access the GUI of an application, all further operations initiated via the GUI should be considered as being performed with the respective application’s rights (see below). This implies that only administrative users may be granted access to the framework administration applications. Access to web-interfaces provided by applications is also performed with the permission to access the applications’ GUIs. An alternative way to remote-access the gateway is the REST interface defined in OGEMA. This interface allows read, write and structure-modification access to the current state of the system (the resource graph) as well as read-access historical data. Communication of the REST messages is performed via https; authentication is via username and password. The access permission can be configured via a set of rules consisting of allow/deny on either data models or specific data nodes, as well as a set of operations that are allowed or denied: reading, writing, creating/deleting, adding sub-nodes and activation/de-activation (the latter is an OGEMA mechanism for making nodes visible to the applications and hiding them from them, respectively).

The REST interface can be used by applications not running on the gateway themselves, such as smartphone applications, or for remote m2m access to the gateway. A special role is played by Javascript-applications that are contained in an application installed on the gateway but run inside the browser of the end-device that the user uses to connect. Such applications can use a one-time combination of a username and a password that is embedded into the html code delivered by the browser and grants data access permissions equal to that of the application. The one-time username/password expires with the user’s session.

Applications running on the gateway can have the same data access permissions as users accessing the gateway via the REST interface (see above). Additionally, the applications can be granted any of the defined Java and OSGi permissions, such as access to the hard drive or access to other OSGi services (usually, applications should be granted neither of these two example permissions, since they are not needed for normal operation and potentially harmful). The check of these permissions is done via the normal Java and OSGi mechanisms, respectively. Applications must define their maximum set of permissions in a permissions.perm file that is contained in every application. Additional constraints may be set during installation or during run-time of the application – the resulting effective set of permissions is the intersection of these two sets. For applications that automatically start with the first gateway start-up, a static configuration file allows to set their initial permissions. In the scope of the SEMIAH project, all applications can be assumed to be pre-installed. So the static configuration of their permissions will be the way to go.

For normal operation mode, there may be no need to access the gateway from outside the building's local network. All relevant information exchange between the gateway and the back-end can be initiated by the gateway. For the user's access to the gateway, it is not clear if a remote access is required or if the user can be constrained to be connected to the local network for access to the gateway (implying the user has to be at home). An additional layer of security might be achieved by configuring the local router or Wifi access point such that external access requests to the gateway are denied.

Future work:

Authentication seems to be only local user/password. This is an area that may need improvement, since SEMIAH should have centrally managed access control ideally supporting multifactor authentication. For user interfaced based on cloud-based services, then existing multifactor authentication solutions of the cloud provider can be utilised.

UIA plans to implement a host-based intrusion detection system for OGEMA which can be used to detect abnormalities and attacks on the home energy management gateway or Demand/Response service. This system will be based on log file analysis from OGEMAs system logs, and will probably run as an OSGi process. This IDS module can probably be utilised also for other purposes, for example as a burglar alarm or fire alarm by monitoring smart plugs and perhaps in the future also custom sensors. This may be used to create auxiliary services around the Demand/Response service, which may improve the business case.

4 Enhancing the Security and Privacy Support of SEMIAH (T8.3)

This section does a high-level description of how the security and privacy of SEMIAH can be enhanced. It first analyses different cloud-based service models for implementing a virtual power plant for SEMIAH. The objective would be that SEMIAH in post project exploitation could be industrialised as a Software as a Service virtual power plant environment. This would allow for dynamically scaling up or down the service according to need in order to improve scalability and resilience as well as reducing costs compared to running such services in-house. We therefore aim at designing security and privacy features compatible with cloud-based best practices where possible.

The section also discusses handling of confidential information in the message oriented middleware that will be utilised by SEMIAH, and also handling of private or confidential information in user preferences. Furthermore, authentication and authorisation models are being analysed, as well as a more general analysis on how to secure the Demand/Response protocol.

Software security and security testing is also being discussed, including how secure design patterns can help us to secure the architecture of SEMIAH. The section finally contains an introduction to the security testing that will be performed in order to detect and report security weaknesses in the SEMIAH products to the other relevant workpackages which can mitigate these in the implementation.

4.1 Cloud Computing Environment

One of the objectives of SEMIAH is being able to run the virtual power plant as a cloud-based service. This has several implications both from a security and privacy perspective which will be investigated in this section. The section discusses SEMIAH from a SAAS, PAAS and IAAS perspective, including related to scalability/elasticity authorisation, deployment etc. It also discusses which authorisation models that are supported by cloud providers, especially to support two-factor authentication.

The cloud is a metaphor for the Internet (or a part of it). The NIST definition of cloud computing, according to NIST SP800-145 is [20]:

“Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.”

Essential characteristics of cloud-based services, is that it is an on-demand self-service with broad network access, so that different thin or thick clients, phones, laptops, workstations, servers, etc. can connect to these services. Cloud-based services provide resource pooling, so that hardware resources can be shared using virtual machines and dynamically assigned on-demand. Such services also provide rapid elasticity, so that the service can scale rapidly up or down on demand. It is also a measured service, so that you typically can have a “pay-as-you-go” subscription, where you only pay for the computing resources that are being utilised.

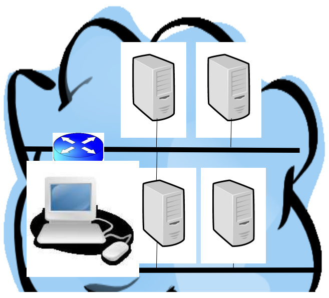


Figure 9: Infrastructure as a Service.

4.1.1 Cloud Service Models

The next three subsections investigate the three basic service models of cloud-based services: Infrastructure as a Service, Platform as a Service and Software as a Service.

4.1.1.1 Infrastructure as a Service (IaaS)

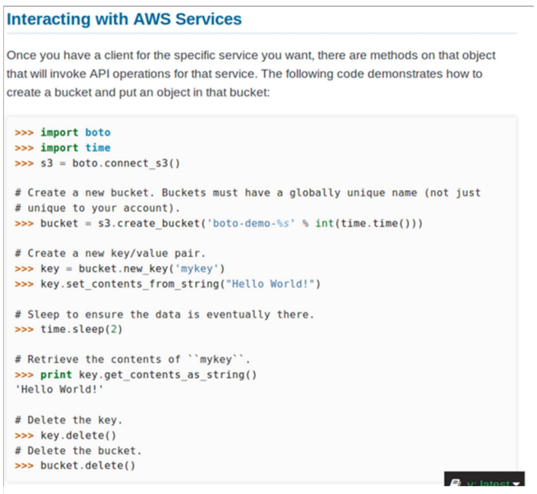
Infrastructure as a Service, as shown in Figure *9*, is the most basic cloud-based service model. This essentially means that you can manage processing, storage, networks as well as run arbitrary software (operating systems or applications) on top of the cloud-based infrastructure.

It is possible to define and control your own virtual infrastructure using IaaS, but you do not control the underlying cloud infrastructure (i.e. the Virtual Machine Monitors - VMM). This also has security implications for SEMIAH, because it means that the virtual power plant operator only can put limited trust in the cloud platform, and will need to protect business sensitive or confidential information that is being stored in the cloud. Also, a vulnerability in the VMM provides a threat to the entire cloud environment.

Microsoft Azure is a third large IAAS player, and provides both Windows and Linux based virtual servers. It uses authentication based on Active Directory/LDAP. It also now supports OpenID authentication and OAuth 2.0 authorisation.

OpenStack (http://www.openstack.org) is a free and open source cloud computing platform initially made to be API compatible with Amazon Web Services. It is being managed by the OpenStack foundation, and was started by Rackspace hosting and NASA. Several large companies have since joined the foundation, including Ericsson, Oracle, several Linux vendors, VMWare and IBM. The objective is to create a standardised IAAS execution environment.

One challenge a Norwegian or Swiss VPP providers might experience with cloud services, is that cloud providers within EU might not send harddisks containing copies of sensitive data to non-EU countries for legal reasons. This happened when the Norwegian company Geodata sent harddisks to Amazon Web Services (AWS) in Ireland in order to copy large amounts of data into the servers. AWS subsequently denied sending the disks back, because Norway was outside EU [21]. Similar scenarios could happen if virtual power plant companies need to back up large amount of cloud stored data in order to have local copies of them. This shows that one must remember that there may be legal issues with cloud-based services, as well as remembering that there actually is some real hardware running in the cloud. It may matter where this hardware is running, especially for critical infrastructures like the Smartgrid, and it may not be trivial to copy or back up large amounts of data via cloud-based services. This is not an issue for the SEMIAH demonstrators, but it could be a risk to consider when the service grows large.



Interacting with AWS Services

Once you have a client for the specific service you want, there are methods on that object that will invoke API operations for that service. The following code demonstrates how to create a bucket and put an object in that bucket:

```
>>> import boto
>>> import time
>>> s3 = boto.connect_s3()

# Create a new bucket. Buckets must have a globally unique name (not just
# unique to your account).
>>> bucket = s3.create_bucket('boto-demo-%s' % int(time.time()))

# Create a new key/value pair.
>>> key = bucket.new_key('mykey')
>>> key.set_contents_from_string("Hello World!")

# Sleep to ensure the data is eventually there.
>>> time.sleep(2)

# Retrieve the contents of ``mykey``.
>>> print key.get_contents_as_string()
'Hello World!'

# Delete the key.
>>> key.delete()
# Delete the bucket.
>>> bucket.delete()
```

Figure 10: Platform as a Service Python example (Amazon Web Services)

A limitation with using IaaS, is that SEMIAH will need to manage the cloud infrastructure itself. There is no easy way to scale the virtual services up beyond the capacity of a single underlying hardware server. It is possible to add a software layer on top of these virtual services that handle redundancy and adds scalability by adding virtual servers, however if this is necessary, then it might be better to consider the next service model: Platform as a Service (PaaS), which inherently supports such scalability.

4.1.2 Platform as a Service (PaaS)

Platform as a Service (PaaS) means that you deploy your own or purchased applications using the cloud providers application programming interface (API). Examples of such interfaces are Amazon Web Services or Google Appengine. Figure *10* illustrates a hello world example for the Amazon Web Services platform using the Python programming language.

An advantage with PaaS is that the platform does all the heavy lifting of abstracting the cloud programming interface from the underlying hardware. The cloud service provider also typically provides some guarantees for the basic security of the underlying platform in the form of a service level agreement (for example detecting and patching software vulnerabilities, detecting attacks on the infrastructure etc.). This means that you program towards virtual cloud resources that are elastic and can dynamically scale up or down according to need. This solves much of the technical problems on how to deploy systems and how to handle scalability that otherwise must be explicitly considered if Infrastructure as a Service is being used. For SEMIAH, PaaS would allow the virtual power plant to scale up or down according to need, without having to consider limitations in the underlying hardware. Designing the VPP scheduling as a PAAS service could for example be one way to ensure that the scheduling scales sufficiently well.

A potential downside is however the pay-as-you-go model, which may be expensive, depending on the business scenario, especially for data-heavy applications sending many messages.

A drawback with PaaS, is that the platforms are not yet standardised, which means that you typically will be locked in to the API of the cloud service provider [22]. This may make it a challenge to move the service to another and possibly cheaper cloud provider, which is not desirable for SEMIAH. A risk could also be that the PaaS provider went out of business. What do you do then? Another challenge with PaaS, is that you then depend on the security model of the cloud provider, which may not be sufficient to the needs of SEMIAH. This means that security controls, for example encryption, will need to be added on top of the virtual power plant application. This means that the application is responsible for its own security, and must take necessary precautions since it runs in an environment that may not be sufficiently trustworthy. It may then be better to use existing solutions or software on Infrastructure level which has a proven security track record, provided that one is willing to take the additional cost of properly managing the security of the different virtual machines.

Amazon Web Services provides its own message queuing service called Simple Queuing Service (SQS). A challenge with this solution is however that it can be quite expensive to run, since it charges money per message, and can also be relatively slow. It supports authentication using access key ID and request signature, or via an X.509 certificate. It does not support basic security, meaning that the messages will need to be encrypted before they are placed in the queue. Users of SQS seem to be plagued both by the cost of the service, as well as it being relatively slow, and not being able to providing any service guarantees, like for example message ordering. There are therefore several guides on the Internet on how to replace SQS using RabbitMQ, which supports the AMQP protocol, which both supports basic security as well as being able to give service quality guarantees on the messaging service. Furthermore each SQS message is limited to 8k of data, which may be a problem. Google App Engine provides a different Task queue interface.

One advantage with PaaS is that it usually ensures some level of data integrity and data persistency, ensuring that data is being backed up several places. Transaction support ensures the possibility to roll back to a previous state. This means that SEMIAH may not need to explicitly manage backups with a PaaS solution, given that the PaaS provider is considered sufficiently trustworthy. It is still possible to back up the entire datastore, if a local backup is desirable. Objects in the datastore (an object-oriented nosql database) will however need to be explicitly encrypted when confidentiality protection is required.

Google cloud SQL is a fully managed MySQL service which encrypts user data at rest and supports encryption of external connections using SSL.

SEMIAH will consist of a mix of SEMIAH specific components, proprietary software modules from Misurio and Fraunhofer as well as hardware units running OGEMA. This means that SEMIAH will not be able to run a pure cloud-based service model. The service model will need to be a hybrid between cloud and own managed devices, as well as potentially a hybrid between own cloud infrastructure and platform-as-a-service, if we decide to use this.

4.2 MOM Confidentiality Handling

The SEMIAH Message Oriented Middleware, for example Rabbitmq, supports both link encryption using Transport Layer Security (TLS) as well as strong authentication. It must however be noted that the MOM itself terminates the TLS links, which means that the MOM can read the messages in cleartext. Furthermore, the MOM will run in the cloud, which means that it cannot be trusted to handle private or confidential information.

This means that private or confidential information will need to be encrypted before it is being sent to the MOM. One way of doing this, is to utilise the Reversible Anonymiser for anonymising and encrypting such sensitive information, given that it is in XML format. The anonymiser does not yet support JSON format, however this may be considered added for SEMIAH if necessary.

4.3 User Preferences

One feature that should support privacy controls is user preferences for SEMIAH. The user should be in control of these preferences, and should be able to update them, and delete them if the user so wishes. The SEMIAH terms of use should also state what a service provider can do with the data, for example whether the service provider can use user preference data or data from the demand/response service in targeted advertising or third party services or not. It is important that the user is in control of data the user owns, and that informed consent is given by the user to access data that may be potentially sensitive from a privacy perspective.

4.3.1 Deployment of Gateway

It is important that the home energy management gateway is easy to deploy, so that a minimum amount of hassle is required, but also that the procedure also so that updating is sufficiently secure, preferably using autoconfiguration. One way to achieve a reasonable level of security is to let the gateway automatically download and install signed and tested software updates from a predefined SEMIAH repository, preferably over a mutually authenticated secure link. Pushing out new updates should be done gradually, possibly via groups of gateways, to avoid unforeseen problems that have not been identified during testing. New updates must furthermore be well tested before they are deployed, and should possibly require co-signing by at least two stakeholders to reduce the risk of faulty deployments of the software.

4.4 Authentication and Authorisation

As mentioned in sections 2.2.3 and 2.2.4, secure authentication and authorisation is essential. When possible, two-factor authentication should be used. Passwords must be stored securely. Secure password involves the enforcement of strong password selection and also the salting of the password hashes.

An important point is that SEMIAH should use a single method, component or system responsible for authenticating and authorising users or services (e.g. a single-sign-on system), instead of relying on device-local authentication. As far as possible, existing standards should be used. A suitable standard for exchange of authentication and authorisation data that supports single sign-on is the Security Assertion Markup Language (SAML) [23]. SAML requires TLS for transport layer security, and communicates over SOAP 1.1 on the back end. Another standard, which is more relevant in a cloud scenario, is OpenID (http://openid.net).

Both SAML and OpenID can interact with the access control policy language XACML [12], and using the ViSPE XACML policy editor [15] described in Section 3.3.1 will allow us to define access policies for SEMIAH in a user-friendly manner.

4.5 Securing the Demand Response Protocol

For the demand and response task, the Smart Energy Profile 2.0 (SEP2) should be used. SEP2 is an international standard for an IP based application protocol specification providing Smart Grid services for home and business energy devices [17].

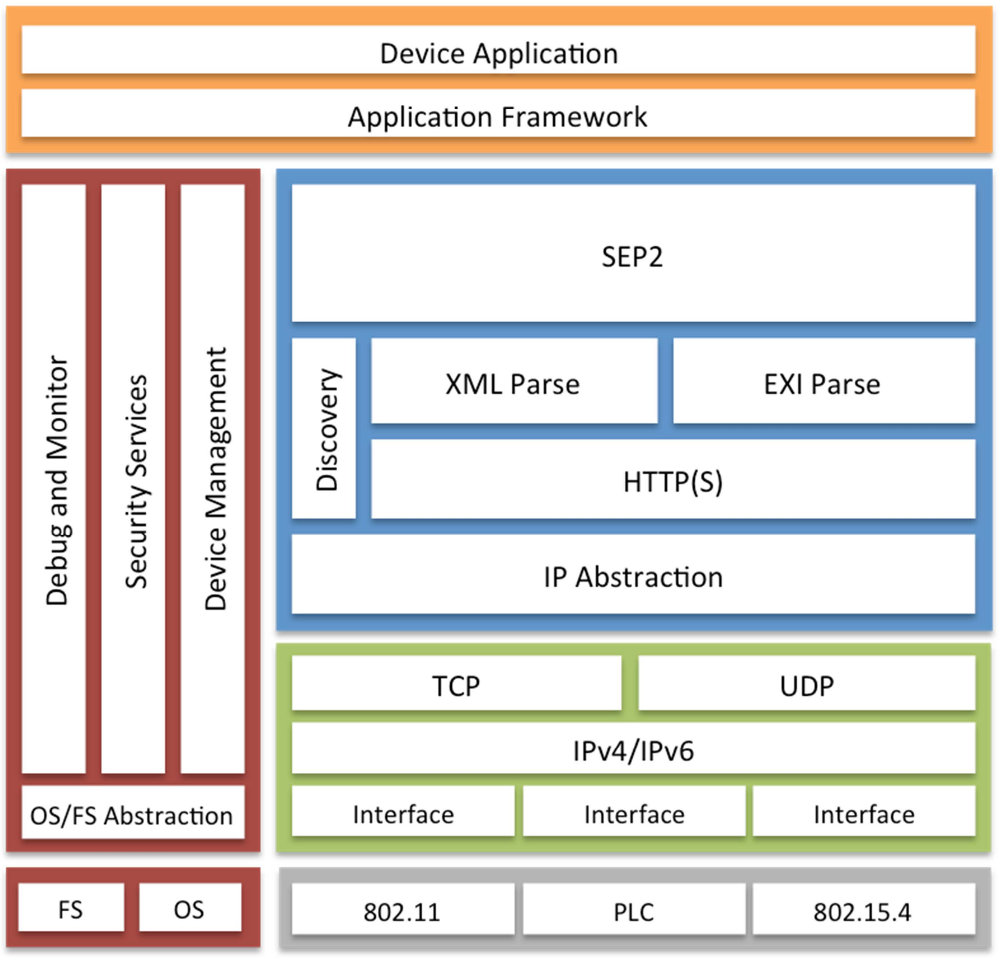


Figure 11: Demand Response using SEP2.

SEP2 has wide range of functionality including pricing, metering, messaging, billing and demand-response load control. It also supports several standards and implementation specific platforms. SEP2 can be used with different interfaces such as Ethernet, Wi-Fi, ZigBee. It uses standard http/https communication methods. SEP2 puts high emphasis on security by using an elliptic cryptography (ECC) based cipher suite.

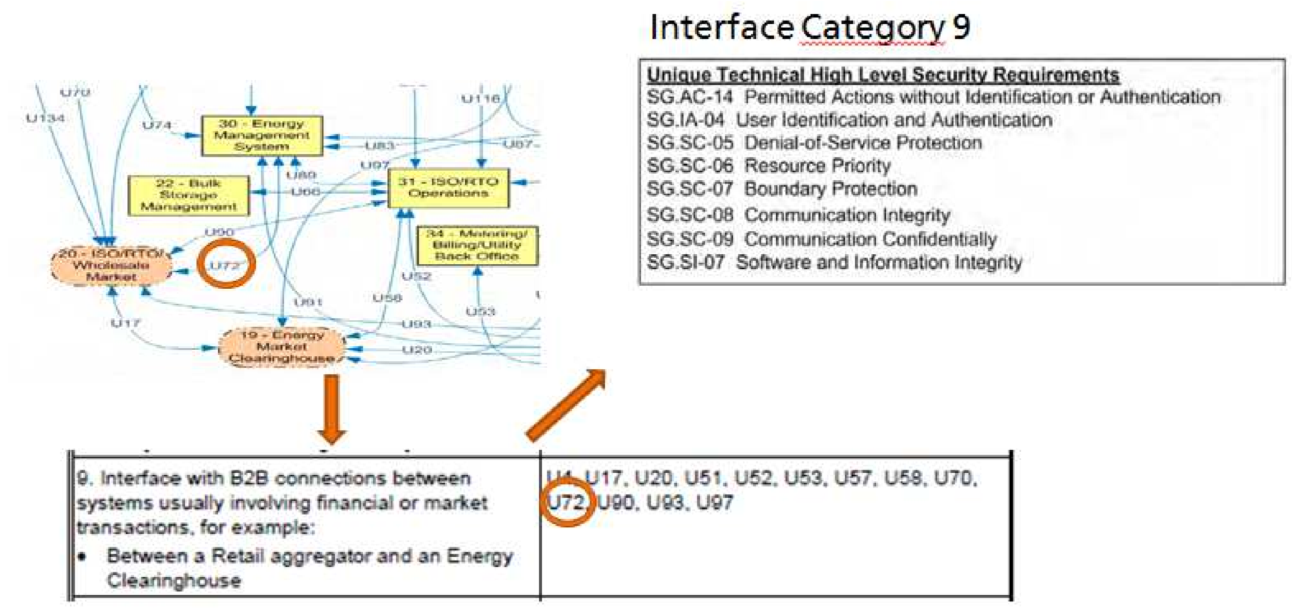
4.6 Software Security and Security Testing

This section describes how software security can be achieved in the SEMIAH project using good practices such as secure design patterns and security testing. Security testing will be coordinated with regression testing activities in WP6, so that introduction of new features, or upgrading of software packages, will ensure that also security tests such as vulnerability scanning will be re-run according to the testing procedures in D6.1. Vulnerability scanning results can be imported into the Verinice risk assessment system based on standardised risk models based on the Common Vulnerability Scoring System (CVSS). This allows for cross-referencing security test results to risks, as described in section 9 of D6.1.

4.6.1 Secure Design Patterns

This section describes more in detail how secure design patterns will be applied in the SEMIAH project. The first volume of NIST IR 7628 describes an approach to identify high-level security requirements. It represents a high-level architecture, followed by a logical reference model. This logical reference model is used to identify and define 22 logical interface categories. For these logical interface categories, so called high-level security requirements are described. All logical interfaces are grouped into these logical interface categories. Logical interfaces in the same interface category have similar security requirements.

Figure 9 shows an example of how to find security requirements for a logical interface within the logical reference model of NIST IR 7628. For example, the interface U72 models the connection from the energy management system to the market. This interface U72 is within the interface category 9. This interface category has the listed requirements.


U134
U70
U74
30 - Energy Management System
U116
U83
U87
U97
22 - Bulk Storage Management
U89
U66
31 - ISO/RTO Operations
U90
20 - ISO/RTO/ Wholesale Market
U72
34 - Metering/ Billing/Utility Back Office
U52
U91
U56
U53
U17
19 - Energy Market Clearinghouse
U93
U20

Interface Category 9

Unique Technical High Level Security Requirements	
SG.AC-14	Permitted Actions without Identification or Authentication
SG.IA-04	User Identification and Authentication
SG.SC-05	Denial-of-Service Protection
SG.SC-06	Resource Priority
SG.SC-07	Boundary Protection
SG.SC-08	Communication Integrity
SG.SC-09	Communication Confidentially
SG.SI-07	Software and Information Integrity

9. Interface with B2B connections between systems usually involving financial or market transactions, for example: • Between a Retail aggregator and an Energy Clearinghouse	U4, U17, U20, U51, U52, U53, U57, U58, U70, U72, U90, U93, U97

Figure 12: Example, to find the security requirements for a logical interface.

A system in the form of SEMIAH does not exist in the logical reference model of NIST IR 7628. Regarding of the example of a virtual power plant in Figure 10, we can extract three parts to model the system. The three parts are [14]:

- **Energy Management System**: is a system of computer-aided tools used by operators of electric utility grids to monitor, control and optimize the performance of the generation and/or transmission system. The monitor and control functions are known as Supervisory Control and Data Acquisition (SCADA); the optimization packages are often referred to as "advanced applications."

- **Distribution Management Systems**: is a suite of application software that supports electric system operations. Example applications include topology processor, online three-phase unbalanced distribution power flow, contingency analysis, study mode analysis, switch order management, short-circuit analysis, volt/VAR management, and loss analysis. These applications provide operations staff and engineering personnel additional information and tools to help accomplish their objectives.

- **Load Management Systems/Demand Response Management System (DRMS):** LMS issues load management commands to appliances or equipment at customer locations in order to decrease load during peak or emergency situations. The DRMS issues pricing or other signals to appliances and equipment at customer locations in order to request customers (or their preprogrammed systems) to decrease or increase their loads in response to the signals.

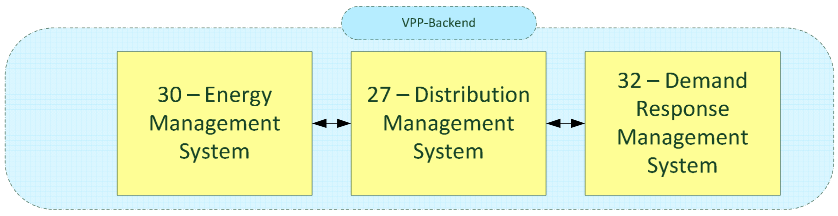

VPP-Backend
30 – Energy Management System
27 – Distribution Management System
32 – Demand Response Management System

Figure 13: Extracted parts to model the virtual power plant of SEMIAH.

To find the relevant security requirements for the SEMIAH backend-system, we have investigated a general example of a system model of a virtual power plant. This example is shown in Figure 11. There exist some definitions about a virtual power plant. Some key points have the following in common:

- A virtual power plant aggregates the capacity of many diverse DERs like generation, load and storage
- The power plants are connected with the virtual power plant as central energy management system via different communication networks
- The virtual power plant creates a single operating profile based on different strategies. The strategy can be based on energy markets, network stability or sales.
- Various data can be involved in order to support the creation of the operating profile. For example different types of prediction are involved.

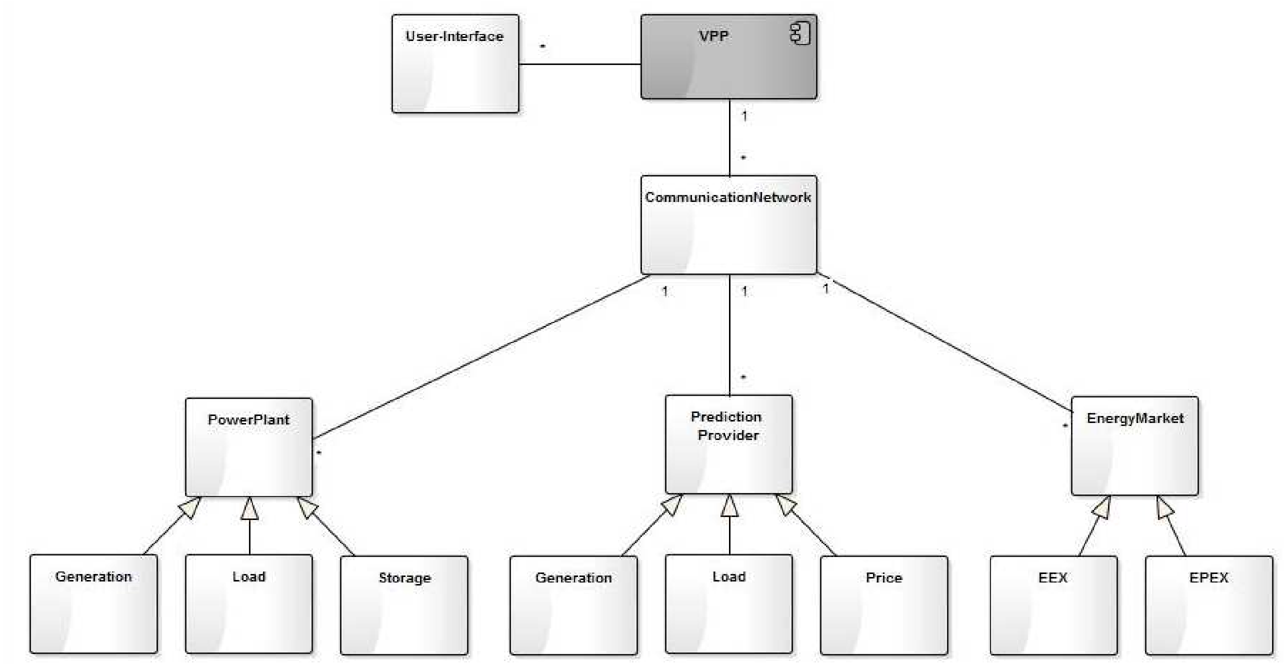


Figure 14: An example of a system model of a virtual power plant, which will be used to derive high-level security requirements from the NIST IR 7628.

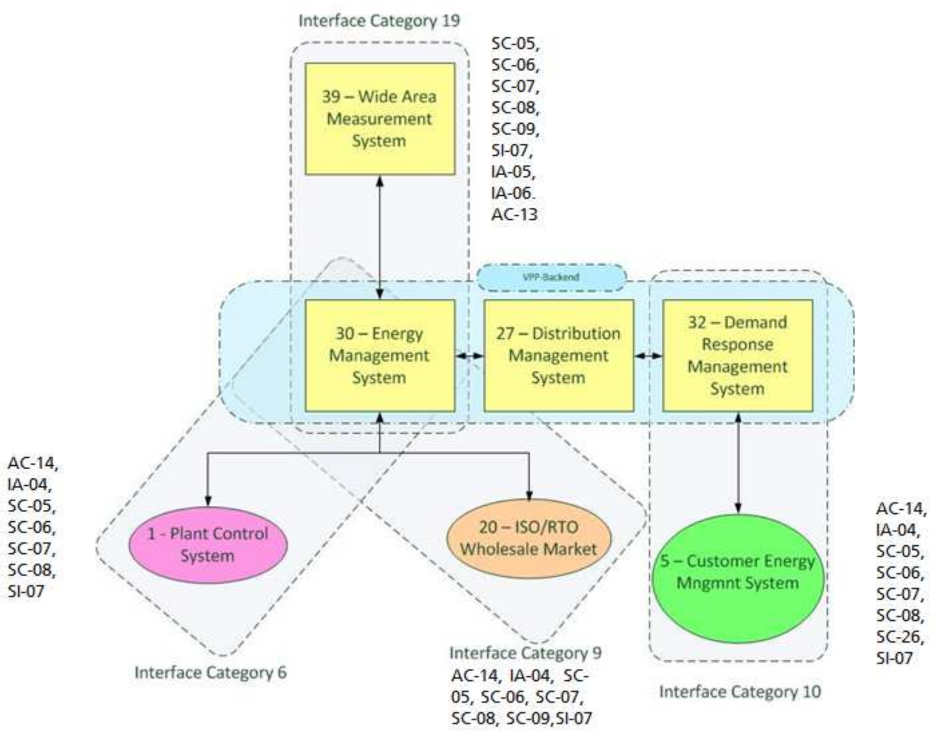

Interface Category 19
SC-05, SC-06, SC-07, SC-08, SC-09, SI-07, IA-05, IA-06. AC-13
39 – Wide Area Measurement System
VPP-Backend
30 – Energy Management System
27 – Distribution Management System
32 – Demand Response Management System
AC-14, IA-04, SC-05, SC-06, SC-07, SC-08, SI-07
1 - Plant Control System
20 – ISO/RTO Wholesale Market
5 – Customer Energy Mngmnt System
AC-14, IA-04, SC-05, SC-06, SC-07, SC-08, SC-26, SI-07
Interface Category 6
Interface Category 9
AC-14, IA-04, SC-05, SC-06, SC-07, SC-08, SC-09,SI-07
Interface Category 10

Figure 15: Mapping system model of virtual power plant to logical reference model (NIST).

An attempt to model the virtual power plant and its connected components with respect to NISTIR 7628 is shown in Figure 11. Four different interface categories were identified:

- Interface 6: Interface between control systems in different organizations
  - Connection to the power plants (if it exists)
- Interface 9: Interface between B2B connections between systems usually involving financial or market transaction
  - Connection to Market (if it exists)
- Interface 10: Interface between control system and non-control corporate systems
  - Connection to households
- Interface 19: Interface between operations decision support systems
  - Connection to prediction provider (if it exists)

For each interface different high level security requirements exist. For example IA-4 means user identification and authentication. These identifiers can be found in the NIST IR. The collected security requirements were summarized and mapped to secure design patterns:

Security Requirement	Secure Design Pattern
Authentication (IA-04)	Patterns for Authentication
Access Control (SC-26,SC-09)	Patterns for Access Control
DoS-Protection (SC-05)	Patterns for Networks (IDS and Firewall), elastic service

Secure-Channel (SC-09,SC-26)	Patterns for Network, Web Services and/or Middleware Security
Monitoring (SI-07)	Security Logger and Auditor

For each category of a secure design pattern, different solutions exist as well as how they can be achieved. Explaining all variants would beyond the scope of this document. Those variations which might fit best for SEMIAH are selected and are shown in the next table:

Security Requirement	Secure Design Pattern
Authentication (IA-04)	Credential
Access Control (SC-26,SC-09)	Attribute-based Access Control, Role based Access Control, Mandatory Access Control
DoS-Protection (SC-05)	IDS (SNORT, OSSEC, SURICATA) and established firewalls
Secure-Channel (SC-09,SC-26)	TLS
Monitoring (SI-07)	Established solutions in the respective programming environment.

4.6.2 Security Testing

Security testing will involve the testing of the Home Energy Management System gateway, the communication between the HEMS gateway and the home devices (e.g. fridge, heating) and the communication between the HEMS gateway and the external server. Smart metering gateway and the Virtual Power Plant (VPP) may also be involved during the security testing. Figure 13 shows possible attack points of the SEMIAH system.

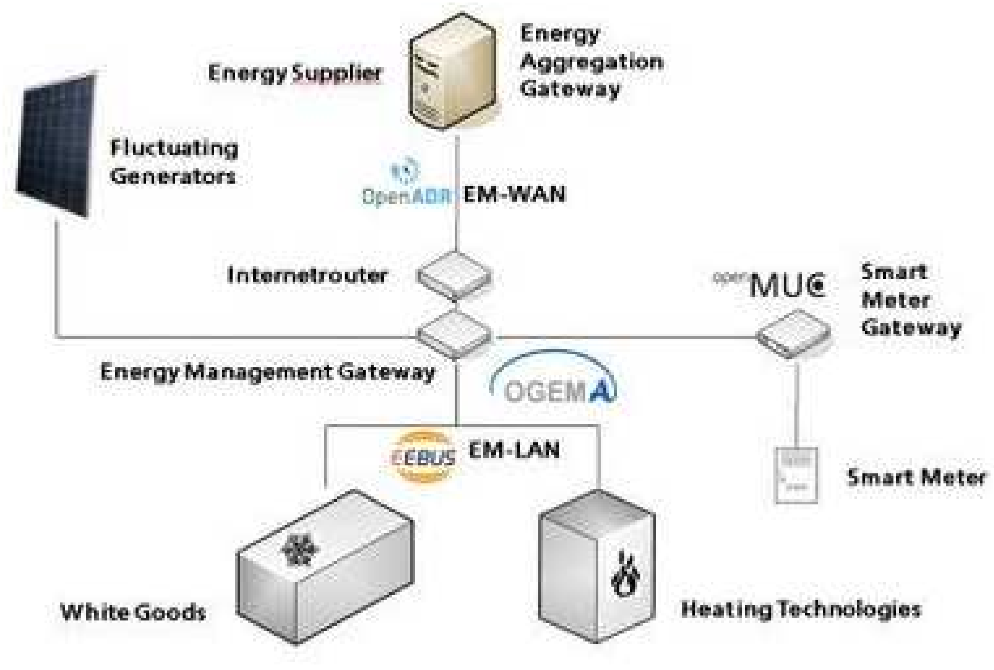


Figure 16: Home Energy Management System Architecture.

The identified attack points of the system are defined according to Figure 13:

1: Home Energy Management LAN

2: Home Energy Management Gateway

3: Smart Meter Gateway

4: External network (Internet) for the communication between the HEMS gateway and the server

5: Mobile app

6: Virtual Power Plant

In order to understand the possible ways of breaking into the system, related to the abuser stories in D3.2, different attack motivations have to be considered such as:

- Fraud by the user of the Home Energy Management system to reduce the energy bill
- Attack by external person on the HEMS with the purpose of information gathering, for example getting info about user customs, sensitive information (which tv program the user watches), time periods when nobody is in the house (robbery), etc.
- Attack by an external person in order to cause damage or annoy the user (e.g. turn of the fridge, increase energy consumption, etc.)
- Attack by an external person on the HEMS gateway to install malware for further combined attacks
- Attack on the Virtual Power Plant to gain information on the users or administrators
- Attack on the Virtual Power Plant to cause relevant impact on the energy supply

4.6.2.1 Test Strategies for Security Testing

Different test concepts for security testing are:

Black box security testing concept: testing method in which the internal structure/design/implementation of the item being tested is not known to the tester.

Gray box security testing concept: testing method in which the tester has some information of the internal structure/design/implementation of the system. The tester itself can be a normal user of a system, or can have some documentation about the system.

White box security testing concept: testing method in which the tester has all the information on the internal structure/design/implementation of the system. He can be a privileged user of a system and/or can have complete documentation about the system.

4.6.2.2 Type of Attacks

Home Energy Management LAN: HEMS LAN provides connection between the HEMS gateway and the devices in the house. As the communication is wireless the attacker can influence the system even from outside the house using the following techniques.

- Blocking traffic
  The attacker keeps sending e.g. deauthentication frames to interrupt the communication between the HEMS gateway and a device. Traffic blocking can also be physical or based on disturbing signals.
- Sniffing the network
  The attacker captures all network data crossing the network from low level packets. If the communication is not encrypted, the attack will able the interpret the communication and collect valuable data.
- Replay attack
  If the attacker manages to capture some network traffic between the HEMS and a device, even if he is not able to obtain any information from that (because of encryption), he will be able to replay the packets, so that the receiver could think, that it originated e.g. by the the gateway. Example: the attacker records the communication when the heating is turned on, later when nobody in the house he replays the scenario from the outside to the turn on the heating again.
- Flooding the device (e.g. heating)
  During the flooding attack the attacker sends an extremely high number of messages to a device so that the device will not be able the process it. Due to this the regular messages will not be processed either.
- Brute forcing attack against the device
  If the attacker has some information on the expected packets he can brute-force the system with different tries. During the brute-forcing the attacker tries several possible passwords for one item e.g. guessing the admin password from a dictionary.
- Weak cryptography attack
  If some encrypted network traffic is captured, the attacker can decrypt the message if the applied cryptography has some significant weaknesses or the attacker has the key for the decryption.
- Fuzzing the communication protocol
  During the fuzzing test the attacker knows the exact format of the messages and tries to send some invalid data to divert the receiver from normal functioning (e.g causing DOS, or applying some type of buffer overflow).

Home Energy Management gateway:

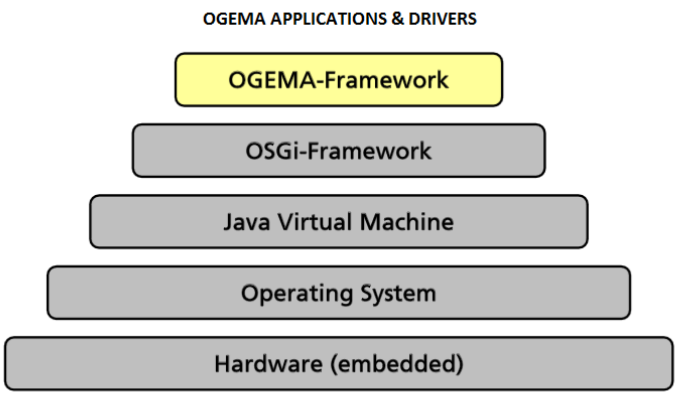


Figure 17: OGEMA gateway layers.

Figure 17 shows the layered representation of the OGEMA home energy management system gateway. The OGEMA gateway is written in Java as a maven project. It is based on the OSGI-framework. OGEMA has a management web interface, where normal as well as admin users can sign in to carry out different tasks. Using OGEMA, users are able to write and apply OGEMA apps, which can carry out different specific tasks with energy management functions. For a connected device OGEMA drivers are necessary to ensure the applicability of the devices by the OGEMA apps. Because of the layered architecture the OGEMA gateway has several dangerous points to test:

Operating system vulnerability

As the gateway software is installed on a traditional operating system (OS), an OS vulnerability can alter the gateway functionality. E.g. if the attacker has system level access to the OS he can write into the OGEMA gateway software process. OS vulnerability can be very simple such as an unnecessary service with factory defaults, or it can be a technically complicated error e.g. in the operating system library. If a new variant of malware infects the system it may even be able to accept commands from a remote command and control server continuously during the attack.

JVM vulnerability (e.g. CVE-2012-0507, etc.)

Using the Java Virtual Environment (JVE) usually ensures secure functioning. As Java is a managed programming language it is more difficult to commit a coding error comparing to native coding. JVE interprets the intermediate code just in time, so it is able the filter dangerous memory corruption errors such as buffer overflows or memory allocation/free errors. However a new zero-day vulnerability in the JVE may be a threat to the gateway as well.

OSGI component vulnerability

OSGI (Open System Gateway Initiative) is a modular service platform for java programming language. Components for OSGI can be remotely installed. OSGI can also contain vulnerabilities itself.

OGEMA-framework vulnerability

One of the key components of the SEMIAH project is the OGEMA framework, which means that and OGEMA apps and drivers security is important. The OGEMA framework is a Java application which provides a web interface for users to carry out its tasks. Therefore the OGEMA system is vulnerable to attacks from a browser using web hacking techniques. The security test of the web interface involves the following types of tests among many others:

brute-forcing of forms and web-server directories, checking of client side and server side validations, checking of configuration errors, checking of information disclosure errors, checking of parameter and session manipulations, etc.

OGEMA app or driver vulnerability

OGEMA apps and drivers can be written for the system. If the programmer lacks appropriate secure coding knowledge he can introduce security vulnerabilities into the system. The security testing will analyse this source of dangers.

External network: the external network is responsible for transmitting the messages between the HEMS gateway and the server. The same type of attacks can be tried here as in the case of HEMS LAN. In the case of HEMS LAN the communication is based on wireless transmission. For the external network the Internet traffic has to be considered. The attacker can be anywhere between the gateway and the server.

- Blocking traffic
- Sniffing the network
- Replay attack
- Flooding the device (e.g. heating)
- Brute forcing attack against the device
- Weak cryptography attack
- Fuzzing the communication protocol

Mobile client: Smart phones are used for configuring the HEMS settings or carry out necessary functions remotely (e.g. turn on the heating remotely half hour prior to the arrival). Smart phones are the main focus of malicious hackers nowadays, so all types of attacks can be considered which provides access to the HEMS gateway through the smart phone for the attacker. Some examples of possible attacks are:

Social engineering attacks

Social engineering can be all kind of hoax to abuse the user using email or web. Social engineering attack can be even shoulder surfing.

Fake app

Fake app is a kind of social engineering attack, when the user downloads an app from an unofficial store. The app can contain malicious side-effects. In the case of HEMS this type of attack can be a very useful and multifunctional HEMS app which has hidden malicious activity (e.g. stealing the users credentials).

Configuration errors

Smart phone apps are typically separated from each other in a sandbox environment to avoid information leakages. However with the configuration files the user can ensure availability to some sensitive data such as the phone book or text messages.

Malware

Malware on smart phones cause more and more trouble to users. This type of attack is important, because it can steal user credentials from the HEMS gateway

Virtual Power Plant

The most important part of security defence is the protection of the Virtual Power Plant (VPP). Attacking the VPP would impact on several households, and might even affect grid stability. The HEMS gateway communicates with the VPP through the internet, therefore the attacker can try to penetrate into the system from any external place. During the attack several techniques can be used:

- Flooding the VPP with huge amount of traffic to carry out Denial of Services (DOS)
- Sending specially crafted packages to cause DOS
- Brute force the system to check mitigation or gain access
- Tampering input parameters to escalate privilege
- Bypass client side validation to gain access
- Manipulating sessions to escalate privileges
- Bypassing server side validation to gain information
- Forcing the system to commit error and obtain information

4.6.2.3 Security Testing Tools

Wireshark

Wireshark is a network analysing tool. It is able to show all traffic that crosses the network card even in promiscuous mode. It presents each layer of the OSI model with all of the necessary data to show. Wireshark has an efficient filtering functionality so that the user can be able to select important data. During the SEMIAH project Wireshark will be used to sniff the network and to control out-coming packets during an attack.

Ettercap

Ettercap is a free and open-source network tool for Man in the middle (MITM) attacks on LAN. With Ettercap the attacker can hijack the traffic so that he can capture data or even to modify the data during the communication of two devices.

Aircrack-ng

Aircrack-ng is a network software suite consisting of a detector, packet sniffer, WEP and WPA/WPA2-PSK cracker and analysis tool for 802.11 wireless LANs. Air-crack ng will be used to crack wireless communication during the project.

Peachfuzzer

Is an interactive fuzzing software used for finding vulnerabilities in communication protocols and software. Peach fuzzer uses a description file to general input data. If the software crashes using a specific input data then more analysis will be provided to find out the vulnerability

Olydbg/Immunity debugger

Ollydbg and Immunity debugger is a win32 user level debugging tool to analyses memory processes. Both of them are dynamic analysers. During the SEMIAH project the software components will be analysed dynamically by these.

Fdgb

Fdbg is a 64 bit debugger for dynamic analysis.

IDA

Interactive Disassembler (IDA) is a *disassembler* for computer software which generates assembly language source code. IDA is able to identify memory segments, exported and imported functions, local variables, calling conventions. With IDA the control flow of the program can be easily analysed.

John the ripper

John the Ripper is a free password cracking software tool. It is able to use several type of password cracking method such as brute force attack, dictionary attack, hybrid attacks. John is programmable and able to use the GPU for brute forcing hashes.

Directory bruteforcer (dirb)

DirBuster (dirb) is a multi-threaded java application designed to brute force directories and files names on web/application servers. Dirbuster has dictionaries for different type of web servers.

Syhant Sandcat browser

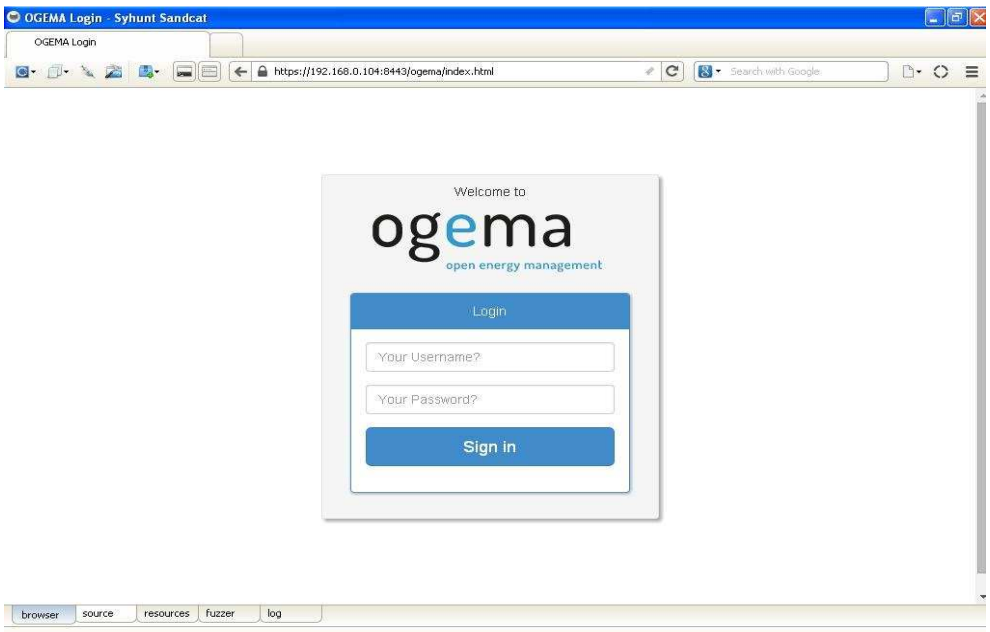


Figure 18: Sandcat hacker browser.

Sandcat is a penetration testing oriented multi-tabbed portable browser. It is well applicable for parameter tampering, simple web-application fuzzing and client side data manipulation.

Browser add-ons (Tamper for Firefox, Web developer extension for Firefox)

Special browser add-ons are good for client side data manipulation during a web application attack.

THC- Hydra (The hackers's choice)

THC Hydra is a fast and flexible Network Login Hacking Tool. It uses a dictionary attack to try various password/login combinations against an Internet service to determine a valid set of login credentials. Hydra will be used to brute-force passwords on web services such as the gateway's login page.

Nmap

Nmap is network security scanner used to discover hosts and services. Nmap is able to determine service versions as well as simple vulnerabilities.

Nikto

Nikto Web Scanner is a Web server scanner that tests Web servers for dangerous files/CGIs, outdated server software and other problems

Sqlmap

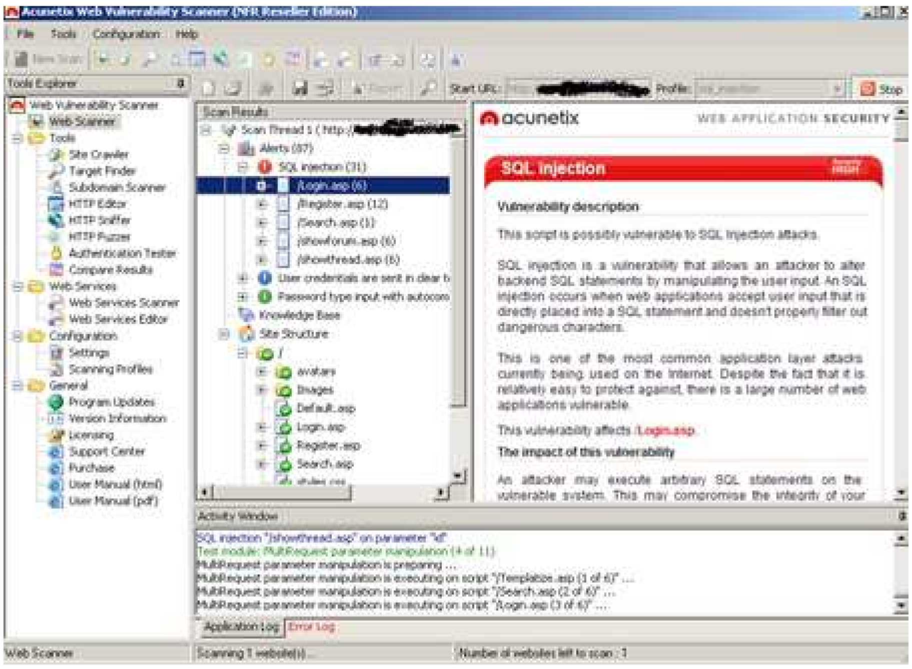
Sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections. Sqlmap will be used for database testing in the SEMIAH project.

OpenVAS

OpenVAS is a framework of several services and tools offering a comprehensive and powerful vulnerability scanning and vulnerability management solution. OpenVAS discover vulnerabilities and categorize them based on its seriousness.

Acunetix

Acunetix, shown in Figure 19, is a popular web vulnerability scanner that contains huge database on web based vulnerabilities. Acunetix is able to discover several type of web vulnerabilities such as XSS, CSRF, SQL injection, hidden directories, session handling problems, encryption errors, information disclosures.



Vulnerability description

This script is possibly vulnerable to SQL injection attacks.

SQL injection is a vulnerability that allows an attacker to alter backend SQL statements by manipulating the user input. An SQL injection occurs when web applications accept user input that is directly placed into a SQL statement and doesn't properly filter out dangerous characters.

This is one of the most common application layer attacks currently being used on the Internet. Despite the fact that it is relatively easy to protect against, there is a large number of web applications vulnerable.

This vulnerability affects /Login.asp.

The impact of this vulnerability

An attacker may execute arbitrary SQL statements on the vulnerable system. This may compromise the integrity of your

```
SQL injection "/showthread.asp" on parameter "id"
Test module: MultiRequest parameter manipulation (4 of 11)
MultiRequest parameter manipulation is preparing ...
MultiRequest parameter manipulation is executing on script "/Template.asp (1 of 6)" ...
MultiRequest parameter manipulation is executing on script "/Search.asp (2 of 6)" ...
MultiRequest parameter manipulation is executing on script "/Login.asp (3 of 6)" ...
```

Scanning 1 website(s) ... Number of websites left to scan : 1

Figure 19: Acunetix web vulnerability scanner.

The security testing will be done periodically during the development and before the deployment of the system. This will be coordinated with other testing activities in WP6.

The documentation of the security test result will contain the following:

- The detailed description of each attack
- Screen shots from the attacks
- List of the experienced vulnerabilities
- Significance of each vulnerability
- Recommendation for fixing the vulnerability

5 Privacy and Security Assessment and Security Monitoring of Demonstrator -

The security and privacy management for the Demonstrator will be based on the standards, methodologies and tools described in Section 3. The privacy and security testing will be coordinated with the test plan in D6.1. Vulnerability testing can for example be used as part of the regression tests to verify whether software vulnerabilities have been patched up or not. Security and Privacy monitoring and management

For risk assessment of the front-end and back-end systems, a MAGERIT-based approach will be taken, as described in Section 3.1.1. The approach will be adapted to the computational power and bandwidth of the respective systems, and in particular, care has to be taken in the front-end system to avoid degrading the performance of the HEMS and its attached devices.

In particular, we may sample different information and metrics from a selection of different HEMS instances to get a representative picture of the state of the front-end systems in cases where the tools and data collection methods are too heavy for implementation on all HEMS.

In general, information from the back-end system and a selection of front-end systems will be collected using tools such as OpenNMS, OpenVAS, suitable Intrusion Detection Systems (IDS) and results from relevant OVAL and OCIL tests.

Verinice will then be used to analyse the collected information for risk analysis support.

5.1 Vulnerability testing and privacy assessment

Vulnerability testing of the implemented system can take place before the release of the system and it can be repeated periodically during the normal operation. It is very important to repeat the analysis regularly, because new vulnerabilities can change the security level of the system drastically. In some cases it may be necessary to modify the system very quickly, e.g. if it turns out that there is a zero day vulnerability in the system, which mean that the system can be compromised. In these cases a new security patch has to be installed or some configuration has to be changed. If a new security hole has not emerged, then the regular vulnerability analysis will focus on new threats that are appeared since the last test.

The vulnerability test can be categorized according to the information or authority the tester has. Similarly to the security testing described in section 4.4, the penetration tester can carry out the test using black box, gray box or white box methodologies. In case of black box testing the tester has no previous information from the system. Using gray box testing the tester has some useful information, e.g. he is a normal user of the system with minimal privileges. In white box testing the tester has administrative rights to carry out his activities.

5.1.1 Vulnerability Test Steps

A vulnerability test consist of the following steps:

1. Information gathering

Information gathering involves all kind of reconnaissance activities when the attacker tries to gain as much information from the target system as it is possible. Information gathering can be passive or active. Passive information gathering involves acquiring information without directly interacting with the target, for example searching for general information using search engines or social networks. In case of active information gathering the tester interacts with the target e.g. by visiting the web interface to gain public information. Information gathering can also be categorized according to the information that the attacker obtains. The information can be general or technical. Technical information for example is the type of services that the target system uses (service obtaining can only based on normal use of the system without e.g port scanners), general information can be anything useful without any technical characteristics (e.g. when was the system developed and who were the developers).

In case of the SEMIAH system this phase involves all of the information gathering activities connecting to SEMIAH such as the project publications, the OGEMA gateway history and general characteristics, user manuals of the system, etc.

2. Network/Software reconnaissance

During the network reconnaissance phase the tester maps the target network. This activity involves the scanning of the network using layer2, layer3 or layer4 protocols. In layer 2 level the available MAC addresses are scanned. Layer 3 level uses the internet protocol, the available devices can be scanned using ICMP messages. In layer 4 level several ways of TCP and UDP scanning can be used. Network reconnaissance also consists of extracting information from the scans and drawing network topology and information flow diagrams of the system.

Software reconnaissance consist of the mapping of the software functionality including input data characteristics and output data characteristics. Every input parameter has to be analyzed later on, so all kind of input and output data has to be listed with its type and intervals.

In case of the SEMIAH system this task involves the scanning of the Home Energy Management System's internal network and also the scan of the services of the virtual power plant. It also involves the OGEMA gateway software mapping.

3. Gaining access/ compromising the system

Gaining access refers to the phase when the attacker finds a vulnerable point where he can penetrate into the system at some level. The attacker can gain access at the operating system level, application level or network level. After the tester has successfully gained access, the next task is to escalate the privileges to obtain complete control of the system.

The penetration can be the result of a weak password, a configuration error or some kind of other weakness. In case of a software the penetration is often available due to memory corruption such as buffer overflow.

System compromising can be any kind of sensitive information gaining or data modification or system availability breaking (DOS). In case of system compromising one or more items from the three information security principals (confidentiality, integrity, availability) are affected.

In case of SEMIAH system compromising, the task is to gain access to the OGEMA gateway or the VPP. Denial of service attacks and modification of the energy bill has to be considered as well.

4. Maintaining access

Maintaining access refers to the action when the attacker tries to retain his ownership of the system. During the maintaining access phase the vulnerability analysis tester checks if a malicious attacker can be hidden in the system after compromising it. Attackers can upload, download or manipulate data, application or configuration. In order to access the system later on, the attacker can open channels, create new user, install a backdoor, etc. Attackers can then use the compromised system to launch further attacks.

In the case of SEMIAH system the task is to analyze if an attacker can open hidden channels for the OGEMA gateway or to the VPP.

5. Covering tracks

Covering tracks is the activity carried out by the attacker to hide malicious acts. It consists of overwriting of system or server data e.g. the log files.

In the case of the SEMIAH system vulnerability testing, the task in this phase is to test if the malicious activity can be detected after a well performed attack.

Vulnerability testing can point out different type of vulnerabilities. A vulnerability can be less serious e.g. information disclosure or even very dangerous e.g. remote code execution. Different type of vulnerabilities are registered in the official vulnerability database (CVE).

Vulnerabilities are classified according to the following groups:

Denial of service: The availability of the system is broken. The system cannot provide its normal operation, so nobody can interact with the system. This can be the result of a very well performed attack with unique input data (e.g tear drop), or simple the consequence of the too many requests that is arriving to the system.

Code execution: The attacker is able to execute his own attacking code by the system. This code can be a server side script executed by the server, a client side script executed by the browser or in some cases the code can be executed by the operating system of the vulnerable software using the compromised process.

Overflow: Overflow vulnerability is a result of the lack of the input validation in case of arrays or strings. If a data is copied to another data which size is smaller and there is no size checking, the second data will overrun. Using this technique the attacker can overwrite important data and modify the behaviour of the target system.

Memory corruption: The memory corruption takes place in the virtual memory where the software code and data are loaded. If the attacker is able to modify a data in the virtual memory, he can modify the program flow for example overwriting the method return address or inflict some exception handling for data.

Sql injection: Sql injection is a database query handling vulnerability. The attacker is able to send own database queries to the system and is able to evaluate them. With sql injection the attacker can acquire the whole database or even system files. In some cases it is also possible to execute own code using the sql injection vulnerability.

Cross site scripting: XSS is a client side vulnerability which affects the client's browser. If a browser input data is not properly validated, then the attacker can send client side scripts (javascript) to the browser. Using client side scripting the attacker can steal session variables or redirect the user to another location.

Directory traversal: this vulnerability is based on the improper validation of filenames. The attacker can place strings in the filenames which leads to directory changing (../attackerfile).

Http response splitting: The attack consists of making the server using carriage return or new line characters in a web response. The result is that the answer is split into two pieces and the second piece contains the attacking code. This technique can be used to e.g. web cache poisoning.

Bypass something: In case of bypass something vulnerabilities some kind of validation is evaded. Bypassing something can lead to privilege escalation.

Gain information: Gain information is the general name of all kind of information disclosure. It can be the version number of a service, source code information, or any useful data.

Cross site request forgery: CSRF is a type of exploitation technique when the user has a valid session to a server and transmits an unintended request which is transacted because of the valid session. The unintended request is the result of a mystification by the attacker.

File inclusion: File inclusion is a vulnerability when the attacker is able to include and upload malicious attacking file with a normal request which is not validated properly.

The CVE database shows that the number of vulnerabilities keep increasing. Figure 17 shows the distribution of different type of vulnerabilities.

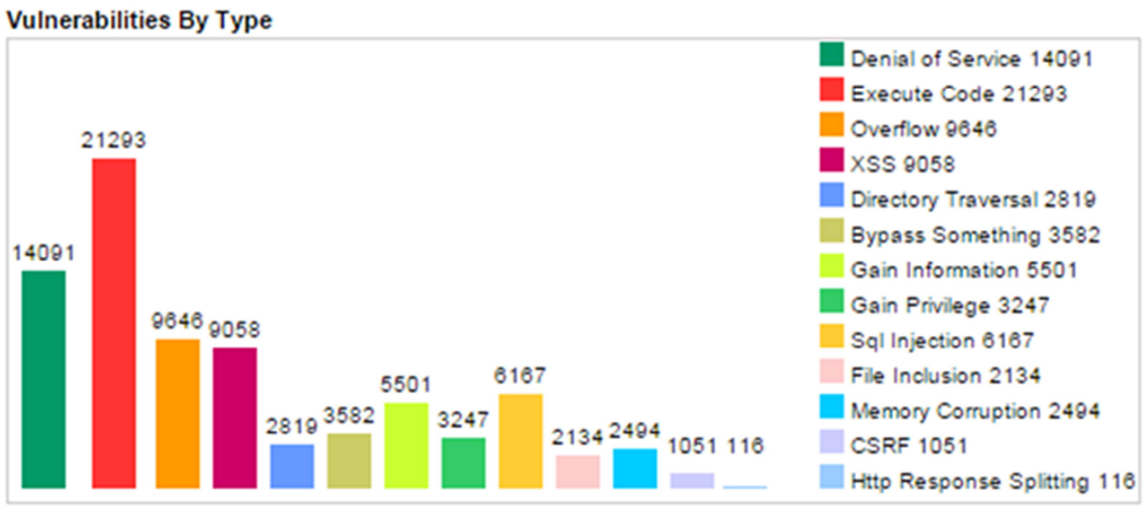


Figure 20 Vulnerabilities by type [18].

5.2 Risk Assessment needs of the Demonstrator vs Full Deployment of SEMIAH

The limited scope for the demonstrator means that some compromises have to be made compared to the full set of methods described in Section 3. One example is that the hardware is a basic ARM9 architecture which lacks support for trusted computing (ARM TrustZone). This ensures that the device cost can be kept as low as possible, whilst still being able to run the OGEMA platform. This tradeoff means that the device will be vulnerable for local attacks, for example key recovery attacks, in order to attack or compromise encrypted services in the virtual power plant. The risk of such attacks is considered small in the demonstrators, which will handle a limited amount of 200 houses. It is believed that monitoring system consistency, for example using file integrity checks and host-based intrusion detection will be sufficient to cover the security needs of the demonstrators.

On the other hand, with a full scale rollout of SEMIAH components that may be in place for up to 20 years, it is important to look forward and take into account the potential for new types of attacks as well as more sophisticated test methods and countermeasures. As more advanced hardware may be deployed for the HEMGs, there are also more opportunities to add advanced IDS systems and vulnerability scanners as well as support for Trusted Computing.

One of the biggest challenges of securing SEMIAH is to determine the risks and threats and plan safeguards and countermeasures up to 20 years in advance. Table *2* summarises the risk assessment needs for the pilot and the full scale roll-out.

Table 2: Risk assessment needs.

	SEMIAH pilot	Full scale roll-out
Assets	One virtual power plant, 200 home gateways and 3-4 controlled devices per gateway. Limited effect on the electricity grid (may affect involved feeder lines for each demonstrator).	Many collaborating virtual power plants controlling millions (or even billions) of home gateways, each with 3-4 controlled devices (or more). May have significant effect on the electricity grid for maintaining grid stability as well as significant effect on the electricity market.

Threats	All currently known threats as well as threats we discover during audits for the software and hardware used by or controlled by SEMIAH. Random threats may occasionally hit the demonstrator. .	Additional threats due to improvements of attack technologies and identification of new vulnerabilities. Another risk is targeted attacks (Advanced Persistent Threats) on the smart grid infrastructure. In a 15-20 years perspective, outdated hardware and software may be an issue. Upgrading to the latest version may then not be possible, so other means (tightly managed firewalls, security monitoring, trusted computing etc) should be used in a full scale rollout to limit the attack surface as much as possible, when patching no longer is feasible.
Impact	Very limited impact to the electricity grid (some feeders only) and no market impact for the demonstrators. An attack may affect individual households.	A targeted attack on a widely deployed SEMIAH infrastructure may have potentially catastrophic impact, causing market manipulation, blackouts and affecting millions of customers.
Safeguards and countermeasures	IDS, managed firewalls and vulnerability analysis.	Continuous safeguard improvement for minimising of the attack surface of outdated devices that by their nature are vulnerable. Trusted Computing may be required to keep the attack surface sufficiently low.

The assets of the newly established pilot system, considering the requirements of the risk assessment – as introduced previously – are now up-to-date, well-defined, and documented. However, technology develops rapidly so the tools in the households will become even more advanced, and the equipment of information science develops as well. So in the future, there will appear new communication protocols together with new applications for the more advanced technical tools.

The possible threats for the pilot system are identified and documented considering all presently possible use-cases. The release of new hardware and software in the future will involve new threats to consider and these have to be predicted as well as possible at the present time.

Different threats will have different relevance for the SEMIAH infrastructure. For the current system, threats are evaluated according to their relevance, but in the future this impact will change as well since new threats may appear with high possible impact. Another factor, which will scale up the impact, is a wide deployment of millions of such energy management gateways worldwide. So already now there have to be considered threats in different impact levels for the demonstrator and for a full deployment of a demand response service.

Safeguards and countermeasures are introduced in the present document - like IDS and vulnerability analysis for the pilot system. For the future, the best safeguard is to minimize the attack surface, as well as introducing improved cryptographic device protection based on trusted computing.

During the threat analysis, one of the primary goals is to determine the acceptable risk level considering the current circumstances. These circumstances to be considered are the attack surface, the vulnerabilities and the resources. Figure *21* shows the interactions of these fundamentals in a 20 years perspective.

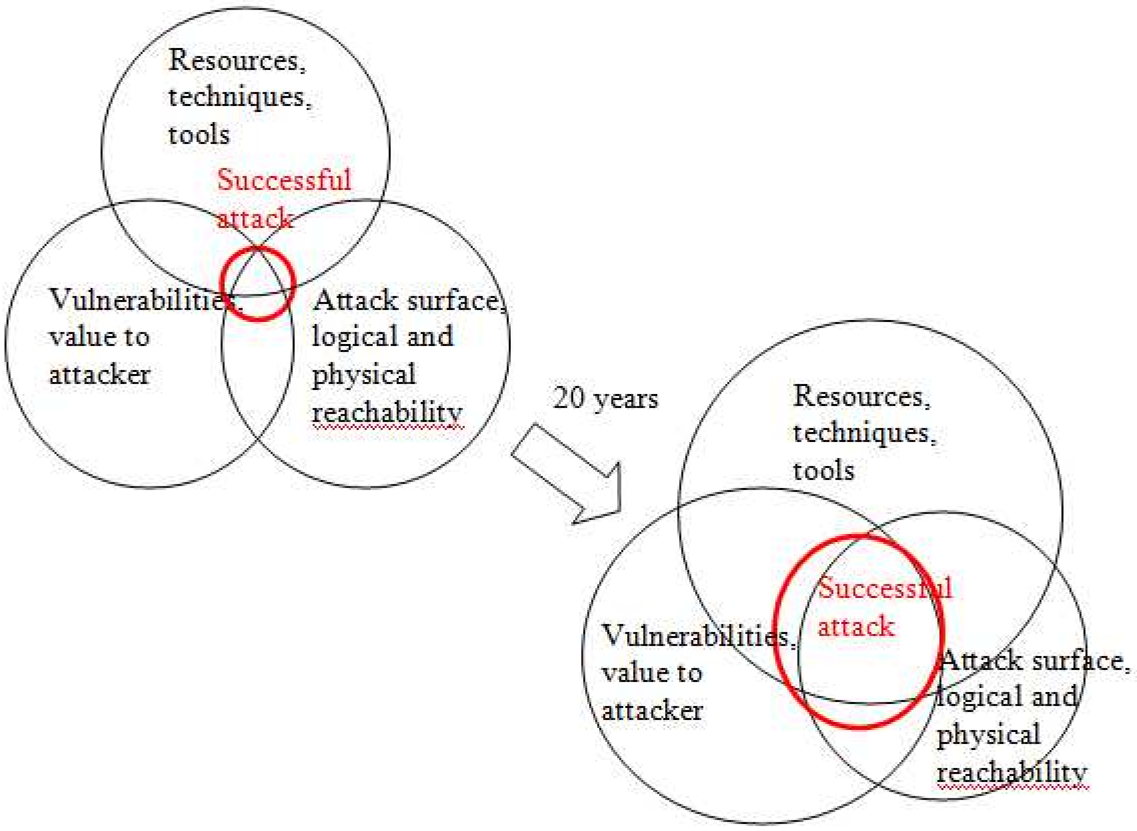

Resources, techniques, tools
Successful attack
Vulnerabilities, value to attacker
Attack surface, logical and physical reachability
20 years
Resources, techniques, tools
Successful attack
Vulnerabilities, value to attacker
Attack surface, logical and physical reachability

Figure 21: The changing probability of a successful attack in 20 years perspective.

Considering the up to 20 years lifetime of the system, the technology development has to be considered as well as the number of the vulnerabilities for the developed system. Another risk is that such a long system lifetime goes beyond the maintenance windows of most software packages used by SEMIAH. To minimise the risk of a successful exploitation, the attack surface of the system has to be kept on a minimum level. Whenever possible, this should be done by ensuring that used software packages are maintained and kept patched and secure. However, this strategy does not mitigate the risk of unknown vulnerabilities, so-called zero-day exploits.

A technology that should be added in future commercial versions of the squid.link gateway to further reduce the attack surface is support for Trusted Computing (e.g. ARM TrustZone), so that a trusted execution environment can be implemented for supporting security critical functions, such as secure memory, encryption, secure boot etc. This would allow for significantly increased device security, especially considering that these devices will be installed and maintained as widely deployed devices with a lifetime of up to 20 years.

5.3 Security and Privacy monitoring

Intrusion detection and prevention and security testing of back-end services and selected front-end services are important aspects of protecting the SEMIAH demonstrator. This, together with applying privacy and security by design, will ensure that SEMIAH is as secure as possible, and that the residual risk after applying necessary countermeasures is considered acceptable.

5.3.1 Intrusion Detection System (IDS)

For the back-end system, hardware is not constrained, so a traditional full IDS system can be used. For this case, privacy-enhanced IDS as described in Section 3.3.3 will be deployed in order to protect the VPP as well as to prevent any leakage of sensitive information in order to protect the privacy of affected users.

When it comes to the front-end system, the situation is different. Running a full IDS on the HEMS would be detrimental to its performance, and instead we opt for a solution based on local and central log analysis. This will be done using e.g. simple OSSEC sensors in the HEMS that will collect relevant log entries add trigger alarms when suspicious activity is detected, and in addition performing correlation analysis for anomaly detection on collected log information centrally to detect suspicious patterns that may e.g. indicate distributed attacks.

6 Summary

This deliverable contains the security and privacy specification of SEMIAH. The objective is to use privacy and security by design to build a system that as far as possible is secure. The system will furthermore handle private or confidential information in an acceptable manner, avoiding unintended leakages of sensitive information. Software security will be ensured using design rules and good practices, and security testing will be applied to verify that the chosen design does not have any unforeseen vulnerabilities. Security monitoring, vulnerability scanning and risk analysis will be performed in order to detect and manage any emerging risks during the demonstrators. Safe and secure deployment of software updates and countermeasures against vulnerabilities will ensure that the system is as resilient and secure as possible within the given project budget. Scalability will be ensured by design decisions that make it possible to scale the virtual power platform service using cloud-based services.

7 References

[1] F. Long, D. Mohlndra, R. C. Seacord, D. F. Sutherland, and D. Svoboda, *The CERT Oracle Secure Coding Standard for Java*. Addison Wesley, 2012.

[2] Cresbo, Gómez, Candau, and Mañas, 'MAGERIT - version 2 Methodology for Information Systems Risk Analysis and Management Book I - The Method', Ministerio de administraciones públicas, Madrid, 2006.

[3] M. A. A. Gómez, J. Candau, and J. A. Mañas, 'MAGERIT – versión 3.0 Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información, Libro II - Catálogo de Elementos'. Ministerio de Hacienda y Administraciones Públicas, 2012.

[4] I. Arce, Kathleen Clark-Fisher, N. Daswani, J. DelGrosso, D. Dhillon, C. Kern, T. Kohno, C. Landwehr, G. McGraw, B. Schoenfield, M. Seltzer, D. Spinellis, I. Tarandach, and J. West, 'Avoiding the top 10 software security design flaws'. IEEE Computer Society, 2014.

[5] European Pariament and the council of the European Union, *Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data.* 1995.

[6] The council of the European Union, 'Directive 2002/58 of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communication sector (Directive on privacy and electronic communications)'. 2002.

[7] A. Cavoukian, S. Taylor, and M. E. Abrams, 'Privacy by Design - Essential for Organizational Accountability and Strong Business Practices', *Identity Inf. Soc.*, vol. 3, no. 2, pp. 405–413, 2010.

[8] N. Ulltveit-Moe and V. Oleshchuk, 'A novel policy-driven reversible anonymisation scheme for XML-based services', *Inf. Syst.*, 2014.

[9] J. Baker, M. Hansbury, and D. Haynes, 'The OVAL Language Specification'. MITRE, 2012.

[10] M. Casipe and C. Schmidt, 'The Open Checklist Interactive Language (OCIL)'. Mitre, 2008.

[11] N. Ulltveit-Moe and V. Oleshchuk, 'Measuring Privacy Leakage for IDS Rules', *Submitted for publication, http://arxiv.org/abs/1308.5421*.

[12] T. (ed) Moses, *OASIS eXtensible Access Control Markup Language (XACML) Version 2.0*. 2005.

[13] A. M. (ed), *OGC 07-026r2 Geospatial eXtensible Access Control Markup Language (GeoXACML) version 1.0*. Open Geospatial Consortium, Inc., 2007.

[14] B. F. H. Debar, D. Curry, *The Intrusion Detection Message Exchange Format (IDMEF)*. 2007.

[15] H. Nergaard, N. Ulltveit-Moe, and T. Gjøsæter, 'A Scratch-based Graphical Policy Editor for XACML', in *ICISSP 2015 Proceedings of the 1st International Conference on Information Systems Security and Privacy ESEO, Angers, Loire Valley, France*, 2015, pp. 182–191.

[16] ENISA Smart Grid Task Force, 'Proposal for a list of security measures for smart grids'. ENISA, 2014.

[17] S. Köpsell and P. Švenda, 'Secure Logging of Retained Data for an Anonymity Service', in *Privacy and Identity Management for Life*, vol. 320, M. Bezzi, P. Duquenoy, S. Fischer-Hübner, M. Hansen, and G. Zhang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 284–298.

[18] N. Ulltveit-Moe and V. Oleshchuk, 'A novel policy-driven reversible anonymisation scheme for XML-based services', *Inf. Syst.*, 2014.

[19] N. Ulltveit-Moe, 'Privacy-enhanced network monitoring'. Universitetet i Agder, 2014.

[20] P. Mell and T. Grance, 'NIST SP800-145 The NIST Definition of Cloud Computing'. National Institute of Standards and Technology, 2011.

[21] M. Jørgensrud, 'Amazon web services - Geodata måtte «smugle» harddisk gjennom Sverige', 2014. [Online]. Available: http://www.digi.no/930720/geodata-maatte-smugle-harddisk-gjennom-sverige. [Accessed: 29-Oct-2014].

[22] E. Hossny, S. Khattab, F. Omara, and H. Hassan, 'A Case Study for Deploying Applications on Heterogeneous PaaS Platforms', in *2013 International Conference on Cloud Computing and Big Data (CloudCom-Asia)*, 2013, pp. 246–253.

[23] S. C. et al, *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard*. 2005.

[24] M. Á. A. Gómez, J. Candau, and J. A. Mañas, ‘MAGERIT – versión 3.0 Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información Libro I - Método’. Ministerio de Hacienda y Administraciones Públicas, 2012.

[25] G. Stoneburner, A. Goguen, and A. Feringa, ‘NIST Special Publication 800-30 Risk Management Guide for Information Technology Systems’. National Institute of Standards and Technology, 2002.

8 Change History

Revision	Date	Responsible	Comment
0.1	05/09/2014	Terje Gjøsæter UIA Nils Ulltveit-Moe UIA	Initial version
0.2	25/09/2014	Terje Gjøsæter UIA Nils Ulltveit-Moe UIA	Added sections to be investigated by Fraunhofer
0.3	21/11/2014	Nils Ulltveit-Moe UIA Terje Gjøsæter UIA	Added sections and analyses from Grimstad meeting.
0.4	08/01/2015	Nils Ulltveit-Moe UIA Terje Gjøsæter UIA	Added security testing methodology.
0.5	12/01/2015	Nils Ulltveit-Moe UIA	Added security and privacy management methodology.
0.6	13/02/2015	Nils Ulltveit-Moe Terje Gjøsæter	Restructuring, added content in various parts-
0.7		Laszlo Erdödi UIA Nils Ulltveit-Moe UIA Stefan Siegl FRAU Terje Gjøsæter UIA	Final touches before second review.
1.0	28/02/2015	Nils Ulltveit-Moe UIA Laszlo Erdödi UIA Terje Gjøsæter UIA Erland Kolstad DEVO	Integrated comments after second review.
1.1	09/07/2015	Terje Gjøsæter UIA Laszlo Erdödi UIA Nils Ulltveit-Moe UIA Rune Hylsberg Jakobsen AAU	Updated according to EU review recommendations as well as an internal review. This includes: Adding an executive summary in the introduction.

		Mohan Lal Kolhe UIA	Elaborated on the privacy-enhanced IDS use case in section 3.3.2, since a figure was missing. Section 3.4.4 describes the relationship between the security testing activities and risk analysis tools and the system and integration test specification in D6.1. Also, suggested importing the grid stability risks from D7.1 into the risk register managed by the Verinice tool. Sections 4.6 and 5 propose that security testing activities are coordinated with regression testing in D6.1. A new section 5.2 was added, describing the risk assessment needs of the demonstrator compared to a full deployment of SEMIAH. Version information was renamed to Change History, elaborated and moved after the references.

9 Appendix A – Security and Privacy Model (from D3.2)

This section describes the modelling of security and privacy in a SEMIAH context.

9.1 Principles and Best Practices in Security

The different aspect of the best practices in security and privacy are presented from the 5-faceted model shown in Figure 18.

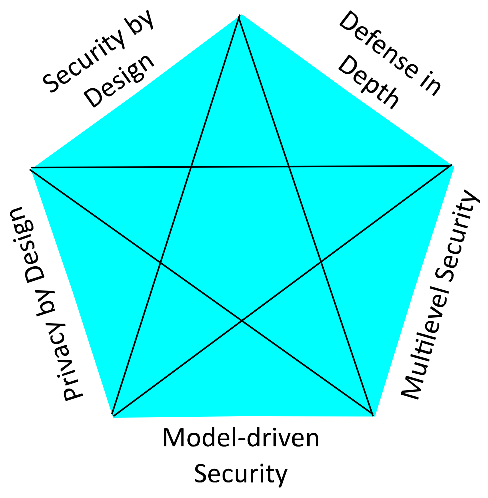


Figure 22 Model for best practices in security and privacy for system analysis.

A brief description of each facet in the model follows in the list below:

- **Security by Design.** Security by design, in software engineering, means that the software has been designed from the ground up to be secure. Malicious practices are taken for granted and care is taken to minimize impact when a security vulnerability is discovered or on invalid user input.
- **Privacy by Design.** Privacy by Design is an approach to systems engineering which takes privacy into account throughout the whole engineering process. The concept is an example of value sensitive design, i.e., to take human values into account in a well-defined matter throughout the whole process and may have been originally derived from this. The concept originates in a joint report on “Privacy-enhancing technologies” by a joint team of the Information and Privacy Commissioner of Ontario, Canada, the Dutch Data Protection Authority and the Netherlands Organization for Applied Scientific Research in 1995.
- **Defense in Depth.** Defense in depth is a concept in which multiple layers of security controls are placed throughout an information technology (IT) system. Its intent is to provide redundancy in the event a security control fails or a vulnerability is exploited that can cover aspects of personnel, procedural, technical and physical for the duration of the system's life cycle.
- **Multilevel Security.** Multilevel security or multiple levels of security (MLS) is the application of a computer system to process information with incompatible classifications (i.e., at different security levels), permit access by users with different security clearances and needs-to-know, and prevent users from obtaining access to information for which they lack authorization.
- **Model-driven security.** Model driven security (MDS) is the tool supported process of modelling security requirements at a high level of abstraction, and using other information sources available about the system (produced by other stakeholders). These inputs, which are expressed in Domain Specific Languages (DSL), are then transformed into enforceable security rules with as little human intervention as possible. MDS explicitly also includes the run-time security management (e.g., entitlements/authorizations), i.e., run-time enforcement of the policy on the protected IT systems, dynamic policy updates and the monitoring of policy violations.

9.2 Coverage

Security should cover all aspects of the system:

- Information security
- Software security
- Physical security
- Hardware security
- Network and communication security
- Cloud services security

Important aspects of information security to cover:

- Confidentiality
- Access control
- Risk management
- Trust
- Resilience
- Integrity
- Availability
- Authenticity
- Non-repudiation

10 Appendix B – Threat model (from D3.2)

The following are some common approaches to threat modelling:

- Attacker-centric (attackers' goals and how to achieve them)
- System-centric (possible attacks for each element of the system)
- Asset-centric (which assets are threatened)

The SEMIAH project will take a multi-faceted approach to threat modelling and start by enumerating important factors including system elements, valuables, attacker types and motivations, possible attacks, and potential attack points.

The important factors of the threat models are:

- System elements / assets
    - o HEMG (OGEMA-based Home Energy Management Gateway)
    - o HEM User interface for Configuration (web/smart phone)
    - o Communication channel in home (ZigBee)
    - o Controlled Home Devices

  - Communication channel HEMG to Backend server (TCP/IP over broadband)
  - Back-end Server (Restful web services HTTP/HTTPS)
  - VPP
  - Market interface module
- Valuables to protect
  - Company confidential info (credentials, configurations, etc.)Sensitive Personal information for users
  - Deep grid / ICT systems
  - Grid Stability
  - Access to Electricity for customers
  - Electricity Company revenue
  - Marketplace
- Attacker types
  - Hacker-as-hobby
  - Hacker-for-hire
  - Bad Neighbour
  - Bad Advertiser
  - Bad competing electricity company
  - Fraud
  - Insider
- Attacker motivations
  - Money (more business for competing company or hacker-for-hire)
  - Free (stolen) electricity
  - “Fun” (hobby-hackers or “script kiddies”)
  - Political reasons (sabotage against opposition, or terrorism)
  - Grudges (bad neighbours)
- Possible attacks
  - See abuser stories
- Potential attack points in the system to be protected
  - User (need to be warned about possible social engineering/phishing attacks)
  - HEMG
  - HEMG user interface
  - Local ZigBee communications
  - Local devices
  - Communication channel HEMG – back-end server
  - Communication channel DSO - backend server

- o Communication channel energy supplier - backend server
- o Communication channel forecast provider – backend server
- o Communication channel third parties – backend server
- o Back-end Server

10.1 ENISA-based Taxonomy of Threats

The proposed ENISA security measures for smart grids contain a taxonomy of threats for smart grid services:

- Natural disaster (fire, flood, thunder, environmental disaster, etc.).
- Damage, loss of IT assets (damage by 3rd party, test corruption, loss of information integrity, loss or destruction of devices, media, documents, media, information leakage)
- Outages (loss of Internet, network, support services, energy, lack of resources, personnel, strike).
- Nefarious activity, abuse/cyber-attacks (ID theft, spam, DoS, malicious code/activity, social engineering, abuse information leakage, rogue certificates, HW/SW manipulation, manipulate information, misuse of audit tools, falsification of records, misuse of information, information systems, unauthorised: access, administration, software installation, software use, compromising confidential information, abuse authorisations hoax, badware, remote activity, targeted attacks).
- Deliberate physical attacks (bomb attack/threat, sabotage, vandalism, theft, information leakage, sharing, and unauthorised physical access).
- Unintentional damage (Erroneous: information sharing/leakage, use or administration of systems/devices, use of unreliable information, unintentional alteration of data, inadequate design, planning/adaptation).
- Failures/malfunction (Device/system failures, disruption of communication links, power supply failure, service provider failure, malfunction).
- Eavesdropping, interception, hijacking (wardriving, intercepting information, man in the middle session hijacking, repudiation of actions, reconnaissance/information gathering, replaying messages).
- Legal (Unauthorised use of copyrighted material, failure to meet contractual obligations, violations of laws).

11 Appendix C – MAGERIT Countermeasures Catalogue

This section presents the complete set of countermeasures described by MAGERIT V3 Book II [24]. This, together with selected elements from the NIST control catalogue [25], can form the basis of the set of controls, safeguards and countermeasures that will be used in SEMIAH.

- General and horizontal safeguards
    - H - General safeguards
    - H.IA - Identification and authentication

- H.AC - Logical access control
- H.ST - Segregation of duties
- H.IR - Incident management
- H.tools - Security tools
- H.tools.AV - Antivirus tool
- H.tools.IDS - IDS / IPS: Intrusion detection / prevention system
- H.tools.CC - Configuration checking tool
- H.tools.VA - Vulnerabilities analysis tool
- H.tools.TM - Traffic monitoring tool
- H.tools.DLP - DLP: Data loss prevention / content monitoring tool
- H.tools.LA - Log analysis tool
- H.tools.HP - Net / honey pot
- H.tools.SFV - Security features verification
- H.VM - Vulnerability management
- H.AU - Logging and auditing

- Safeguards to protect data/information
  - D – Information protection
  - D.A – Backups of data
  - D.I - Integrity assurance
  - D.C - Encryption of information
  - D.DS - Use of electronic signatures
  - D.TS - Use of electronic dating services (time stamping)
- Safeguards for cryptographic keys
  - K - Cryptographic keys management
  - K.IC - Cipher key management information
  - K.DS - Digital signatures management
  - K.disk - Key management for cryptographic containers
  - K.comms - Key management for communications
  - K.509 - Certificate management
- Safeguards to protect services
  - S - Service protection
  - S.A - Ensuring availability
  - S.start - Acceptance and putting into operation
  - S.SC - Apply security profiles
  - S.op - Operation

  - S.CM - Management of changes (upgrades and replacements)
  - S.end - Termination
  - S.www - Securing services and web applications
  - S.email - Email protection
  - S.dir - Directory protection
  - S.dns - Protection of the domain name server (DNS)
  - S.TW - Telecommuting
  - S.voip - Voice over IP
  - S.SCADA – SCADA traffic
- Safeguards to protect applications (software)
  - SW - Protection of computer applications
  - SW.A - Backups
  - SW.start - Put into operation
  - SW.SC - Apply security profiles
  - SW.op - Operation / production
  - SW.CM - Changes (updates and maintenance)
  - SW.end - Termination
- Safeguards to protect equipment (hardware)
  - HW - Protection of information technology equipment
  - HW.start - Put into operation
  - HW.SC - Apply security profiles
  - HW.A - Ensuring availability
  - HW.op - Operation
  - HW.CM - Changes (updates and maintenance)
  - HW.end - Termination
  - HW.PCD - Mobile computing
  - HW.print - Reproduction of documents
  - HW.pabx - Protection of switchboard (PABX)
- Safeguards to protect communications
  - COM - Communications Protection
  - COM.start - Entry into service
  - COM.SC - Apply security profiles
  - COM.A - Ensuring availability
  - COM.aut - Channel authentication
  - COM.I - Protecting the integrity of data exchanged

  - COM.C - Cryptographic protection of confidentiality of data exchanged
  - COM.op - Operation
  - COM.CM - Changes (updates and maintenance)
  - COM.end - Termination
  - COM.internet - Internet use / access
  - COM.wifi – Wireless security (WiFi)
  - COM.mobile - Mobile phones
  - COM.DS - Segregation of networks into domains
- Safeguards at the interconnection points with other systems
  - IP - Interconnection points: connections between trusted zones
  - IP.SPP - Perimeter protection system
  - IP.BS - Protect border system
- Safeguards of information support
  - MP – Information storage protection
  - MP.A - Ensuring availability
  - MP.IC - Cryptographic protection of content
  - MP.clean - Cleaning of content
  - MP.end - Media destruction
- Safeguards of auxiliary elements
  - AUX - Auxiliary elements
  - AUX.A - Ensuring availability
  - AUX.start - Installation
  - AUX.power - Power supply
  - AUX.AC - Climate
  - AUX.wires - Wiring protection
- Physical security – securing facilities
  - L - Protection of facilities
  - L.design - Design
  - L.depth - Defense in depth
  - L.AC - Physical access control
  - L.A - Ensuring availability
  - L.end - Termination
- Safeguards for personnel

For personnel that are connected to the information system.

  - PS - Personnel management

- PS.AT - Training and awareness
- PS.A - Ensuring availability

- Safeguards for organizational type

For those relating to governance of security.

- G - Organization
- G.RM - Risk management
- G.plan - Planning security
- G.exam - Safety inspections

- Continuity of operations

Prevention and response to disasters.

- BC - Business Continuity
- BC.BIA - Business impact analysis (BIA)
- BC.DRP - Disaster Recovery Plan (DRP)

- Outsourcing

The border between the security services provided internally and contracted services to third parties is increasingly flexible. In these cases it is important to consider these aspects of contractual relationship:

- E - External Relations
- E.SLA: service level agreement, if availability is a value
- E.NDA - confidentially, if confidentiality is a value
- E.I - Identification and qualification of personnel
- E.E - Procedures for escalation and troubleshooting
- E.T - Termination procedure (duration in time of the responsibilities assumed)
- E.R - Assumption of responsibilities and penalties for breach
- E.1 - Agreements for exchange of information and software
- E.2 - External access
- E.3 - Services provided by other organizations
- E.4 - Personal outsourced

- Acquisition and development
  - NEW - Acquisition / development
  - NEW.S - Services: acquisition or development
  - NEW.SW - Applications: acquisition or development
  - NEW.HW - Equipment: acquisition or development
  - NEW.COM - Communications: acquisition or contracting
  - NEW.MP - Information storage systems: acquisition
  - NEW.C - Products certified or accredited