Project Acronym: CUMULUS
Project Title: Certification infrastrUcture for MUlti-Layer cloUd Services
Call identifier: FP7-ICT-2011-8
Grant agreement no.: 318580
Starting date: 1st October 2012
Ending date: 30th September 2015

D6.3 Smart Cities Pilot

AUTHOR(S): Antonio Álvarez (WELL), Robin Jollans (WELL), Luca Pino (CITY), Filippo Gaudenzi (UMIL)

REVIEWER(S): Antonio Maña (UMA), Rajesh Harjani (UMA), Hristo Koshutanski (UMA)

Date 05/12/2014

Summary

DOCUMENT HISTORY **6**
EXECUTIVE SUMMARY (WELL) **7**
1. INTRODUCTION **8**
2. SMART LIGHTING SCENARIO **9**
2.1. Specifications 9
2.1.1. Energy Management *10*
2.1.2. Maintenance and troubleshooting *10*
2.1.3. Equipment inventory *10*
2.1.4. Reporting *11*
2.2. Scenario architecture 11
2.3. Security mechanisms 12
2.3.1. Using TPM to encrypt files *15*
3. CERTIFICATION OF SECURITY PROPERTIES **18**
3.1. Work planned 18
3.1.1. External data exchange confidentiality *20*
3.1.2. Network authenticated server access *28*
3.1.3. Tenant Isolation Level *29*
3.1.4. Confidentiality Data Access Level *30*
3.1.5. Data Leakage Prevention *32*
3.2. Other ToCs and properties under research 33
3.2.1. Data leakage prevention *33*
3.2.2. Confidentiality data access level *35*
3.2.3. Storage Freshness *37*
3.2.4. Data Durability *38*
3.2.5. Software Integrity Protection *38*
3.2.6. Measures Integrity *39*
3.2.7. Availability of information (Quality of Service) *39*
3.2.8. Non-repudiation of origin *40*
4. CONCLUSIONS AND NEXT STEPS **40**
REFERENCES **41**

List of Figures

Figure 1. WeLight cloud infrastructure 12
Figure 2. Monitoring infrastructure 20
Figure 3. Testing infrastructure 20
Figure 4. Event caption and sending through the Event Bus 21
Figure 5. Event generated when receiving a packet 24
Figure 6. Event generated when sending a packet 25
Figure 7. Main steps of the monitoring process for certifying the correct operation of the VPN 26
Figure 9. NMAP results. Relevant information is highlighted 28
Figure 8. Secure Communications 29
Figure 10. Flow for database verification of grants 31
Figure 11. Flow for system verification of access policy 31
Figure 12. Multilayer Certification. Data Leakage Prevention 32
Figure 13. DB Encryption Test Flow 33

List of Tables

Table 1. Security properties identified in D6.1 14
Table 2. Fulfillment of MSR´s by SM´s 14
Table 3. Mapping of SP´s by SM´s 15
Table 4. Files and directories used in the process of encrypting files by means of TPM 16
Table 5. Installing software locations 17
Table 6. Security properties defined for Smart Cities pilot 19
Table 7. Web Server electrical panel 34

Acronyms

ACL	Access Control List
APN	Access Point Name
CA	Certification Authority
ESCO	Energy Service Company
FW	Firewall
LCU	Light Control Unit
LUKS	Linux Unified Key Setup
MBR	Master Boot Record
MSR	Main Security Requirement
ORM	Object/Relational Mapping
RHEL	Red Hat Enterprise Linux
SLES	SUSE Linux Enterprise Server
SP	Security Property
SSH	Secure Shell

TCG	Trusted Computing Group
ToC	Target of Certification
TPM	Trusted Platform Module
TSS	Trusted Software Stack

Document History

Revision	Date	Author	Modifications
V0.0	08/08/2014	Antonio Álvarez	Template
V0.0.1	01/09/2014	Antonio Álvarez	Some changes on the template
V0.0.2	04/09/2014	Antonio Álvarez	Some changes on the template
V0.1	04/09/2014	Antonio Álvarez	Writing introduction
V0.1.1	16/09/2014	Antonio Álvarez	Changes in document structure
V0.1.2	22/09/2014	Antonio Álvarez	Changes in document structure
V0.1.3	07/10/2014	Antonio Álvarez	Writing section 3
V0.1.4	12/10/2014	Antonio Álvarez	Writing section 3
V0.1.5	10/11/2014	Antonio Álvarez	New structure for the document. New contents
V0.1.6	11/11/2014	Antonio Álvarez	New contents for section 3
V0.1.7	19/11/2014	Antonio Álvarez	New contents for section 3
V0.2	21/11/2014	Antonio Álvarez	Writing conclusions and next steps
V0.3	04/12/2014	Antonio Álvarez	Version taking into account UMA review
V0.4	05/12/2014	Antonio Álvarez	Version integrating UMA second review
V1.0	05/12/2014	Antonio Álvarez	Final version

Executive Summary (WELL)

Deliverable D6.3 provides the Smart Cities Pilot and its integration with the CUMULUS framework testbed. This pilot is a SaaS solution for management of public lighting. The deliverable covers the implementation of several security mechanisms which make the product more secure and in consequence more marketable, standing out from the competence. The deliverable also covers the process of certifying the security properties provided by these mechanisms. The ongoing work is reported and the next steps to be taken are finally explained. Along with this deliverable, a couple of confidential annexes are provided including the user guide and deployment guide for the pilot.

1. Introduction

This deliverable provides the documentation related to the public lighting scenario used as a pilot case for the CUMULUS project. This scenario is based on WeLight, a commercial application designed to help to decrease the energy consumption of lampposts in municipalities. The certification services provided by the CUMULUS framework are being tested against this pilot as well as the eHealth pilot, whose related deliverable is D6.4.

The CUMULUS framework intends to be an instrument that helps to build trust in cloud services, like WeLight. In an automatic manner, the tools provided by CUMULUS can certify the fulfilment of particular security properties. Prior to the certification of these properties, the corresponding implemented security mechanisms must be in place. Therefore, the benefits the particular scenarios obtain from CUMULUS are twofold: 1) on the one hand, the product is more secure after the implementation of such mechanisms and 2) CUMULUS guarantees the related security properties, what undoubtedly makes the product more appealing and marketable.

In D6.1 the application of this scenario as a use case of the CUMULUS framework was studied in depth. As a result, a list of Main Security Requirements (MSR) that had to be demanded to the pilot was specified. Besides, a set of security properties (SP), defined at a functional level was also specified. Another very important result was the specification of functional and security requirements for the framework itself. In fact, these requirements and their fulfillment will be checked in D6.5 and D6.6 as a part of the validation process.

On the other hand, in D2.1 a catalogue of security properties formally specified is provided. Basing on the aforementioned catalogue, in D3.2 it is specified the list of security properties to be certified on the smart cities pilot. So far only single and multilayer certificates have been considered, however, D3.3 will deal with incremental and hybrid certification and new security properties will be proposed as proofs of concept of these certification methods.

It is of paramount importance to distinguish the implementation of security mechanisms improving the security of a cloud service and the certification of the corresponding security properties. The former is the prerequisite for the latter.

The document follows the structure specified below:

Section 2 deals with the Smart Lighting Scenario. WeLight is presented as an innovative product created to solve the long-standing problem of the management of the public lighting in Spain. Firstly, the specifications of the product are presented so as to provide the reader with a complete overview of it. Secondly, the architecture is briefly explained highlighting the fact that WeLight is a SaaS product. Later on, the security mechanisms implemented up to now are briefly explained and, furthermore, they are related to the MSR´s and the SP´s defined in D2.1 so as to give a complete vision of the ongoing work. The use of TPM to encrypt files and the work carried out to include this security mechanism in WeLight has been considered interesting enough to be discussed more in depth.

Once the security mechanisms are explained, the process of certifying the related security properties is detailed in Section 3. Within the scope of using single and multilayer certificates, a work plan including six security properties has been defined. Testing and monitoring techniques, using TC Proofs in some cases, are used. The infrastructure to carry out both testing and monitoring process is described. The technical work accomplished to get these properties certified is explained on detail. All of this is covered in Subsection 3.1.

Nevertheless, other security properties are under research and in such sense meaningful progress are being made. With the purpose of reflecting that ongoing work, subsection 3.2 provides further information, even giving away some details about discussions considered relevant.

Finally, in section 4 the conclusions obtained from the content of the deliverable are reflected and the next steps are briefly described.

Besides, this deliverable has also a couple of annexes with both the deployment and the user guides. This part of the deliverable is **confidential** and an IPR statement is provided in such sense.

2. Smart Lighting Scenario

The management of public lighting in Spain has traditionally presented several problems that have turned out to be very difficult to solve. The inefficient electrical installations have entailed high economical and energetic costs for families, companies and public institutions. This problem could be faced by providing good management skills but people working in public institutions have shown limited management abilities in this sense. On top of that, the lack of investment to improve the knowledge in public administration with the goal of bringing a more efficient management of electrical power, along with the constant increases in the price of electrical energy, have worsened this scenario more and more.

Given the described scenario, it is foreseeable that any solution aiming at diminishing the electrical consumption must be very welcome. Public administrations, lamp makers or even ESCO´s (Energy Services Company) are likely clients for these solutions. Public administrations themselves and energy consultants could recommend them.

As a matter of fact, by analysing the business model followed by ESCO´s, it is relatively straightforward to conclude their interest on solutions like this. An ESCO usually gets the long term concession of street lighting (10-15 years) for a lower price than what the city council currently pays. Once contracted, the ESCO will make investments to reduce the electricity bill and the profit will be based on achieving a reduction of that bill under what the city council pays them. As saving is key in their business model, they will become very interested in the detection of real time deviations in consumption. The cost of contracting this kind of solutions will be small in comparison to the saving obtained in the bill, what makes worthwhile to rely on this service.

2.1. Specifications

WeLight

WeLight is a product designed to manage public lighting. It is an open and adaptable architecture used to control and monitor any public lighting panel. In this sense, *open* means that there are several standard interfaces available in the Light Control Unit (LCU) that allow interacting with several devices that may exist within an electrical panel. Some of the devices can be: network analyzers, flux regulators, contactors or peer-to-peer telemanagement systems, to name but a few. On the other hand, *adaptable* means that 1) The system can allow integrations with third-party systems, either by means of webservices, or directly by means of the LCU. 2) There are several setup options. For example it is possible to choose between wired or wireless (GPRS) communications.

WeLight features were specified in D6.1 (M6). Below we provide an updated description of these features according to the last released version of the product.

2.1.1. Energy Management

Real time consumption monitoring and public lighting scheduling. Features are listed below:

- Daily, monthly and yearly consumption analysis by using both electrical and economic variables
- Saving measures verification. Provision of comparatives
- Information about electrical measurements per phase in customizable time intervals.
- Billing creation and configuration.
- Consumption simulation
- On and Off schedule management
- Control of head-end regulators and other intervention and saving devices

2.1.2. Maintenance and troubleshooting

Reception of alarms due to breakdowns and blackouts. Incident monitoring and management:

- Alarm management both at the electrical panel and particular phases.
- Customizable configuration to activate and deactivate alarms.
- Customizable configuration of thresholds to activate alarms.
- Alarm notification configuration by means of e-mail and mobile app.
- Possibility of integration with other systems owned by the client.

2.1.3. Equipment inventory

Asset management by using geolocation and key parameters:

- Panel and luminaires inventory
    - Graphical geolocation
    - Panel status
    - Existing equipment within the panel.
    - Panel electrical configuration

- Consumption, service level, inventory and deviations reports.

2.1.4. Reporting

Consumption, savings, service level, inventory, saving deviations and billing reports.

- Consumption reports
- Savings report
- Alarm reports
- Data and electrical variables analysis
  - Bill calculation
  - Calculation of graphics representing power
- Comparation between forecast and real data.
- Detailed reports and executive summaries
- Exportable into PDF and Excel

2.2. Scenario architecture

The system is composed of a hardware unit, the LCU that manages:

1) The physical equipment responsible for obtaining the measurements
2) The switch-on and switch-off of the panels
3) The communications with the server
4) The control of the additional elements existing in the panel (like flux regulators, for instance), and finally a network analyzer that allows to collect electrical measurements.

This solution can be installed on the cloud, as a SaaS product, on Wellness Telecom cloud infrastructure. However, the client has also the chance to install it on his/her premises if required. The communications are usually mobile (GPRS), given that the panels are geographically scattered, although the solution is able to work with any communication network already operating.

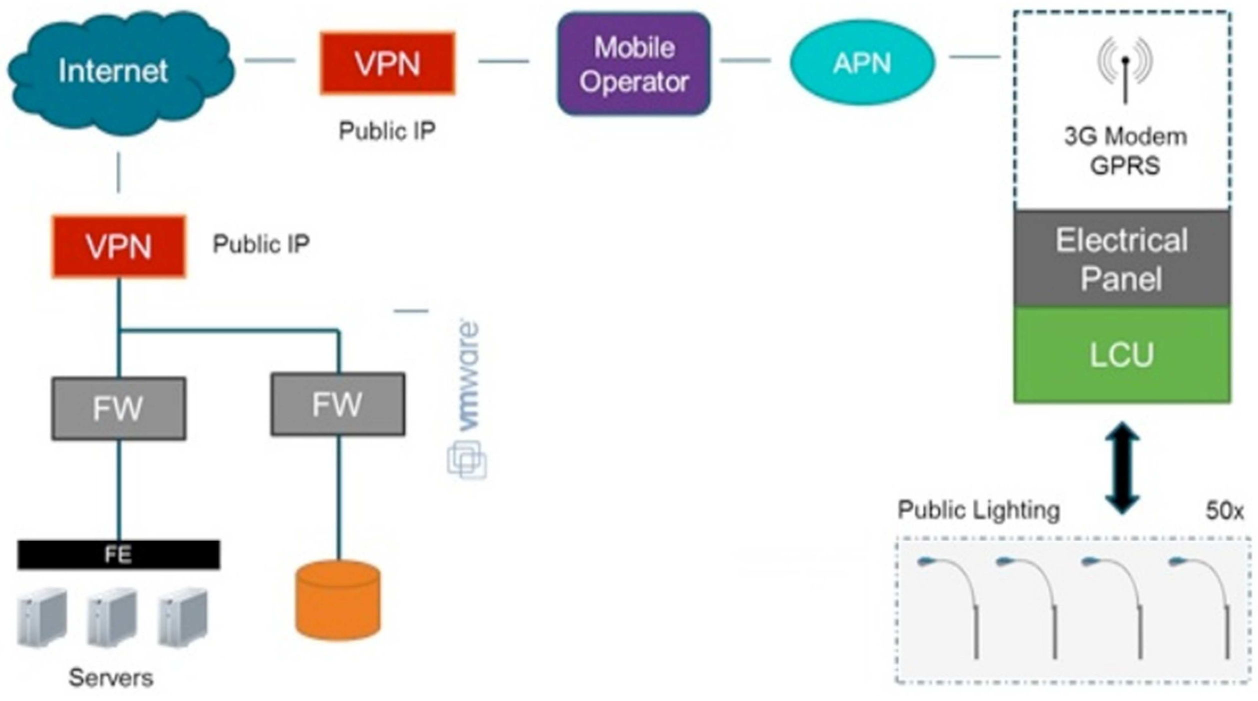

Figure 1. WeLight cloud infrastructure

The figure above illustrates the way WeLight is deployed using a cloud infrastructure:

- On the one hand, LCU´s collect the consumption measurements from a group of lampposts.
- On the other hand, the information is sent by means of GPRS or 3G, passes through the Internet and is finally stored on the cloud infrastructure (provided by Wellness Telecom or the client themselves).

2.3. Security mechanisms

This section provides a summary of the security mechanisms that have been implemented in the context of CUMULUS.

The relation of these security mechanisms with the main security requirements and the security properties that were specified at high level in D6.1 [13][14] will be discussed. First, the MSR´s are reviewed. Second, the SP´s identified in D6.1 are summarized. Finally, the new security mechanisms developed in the context of CUMULUS and those which were already in place are briefly explained and their relation with the MSR´s and SP´s is identified.

The MSR´s defined in D6.1 are listed below:

- MSR.001: *The system must be able to verify a user´s credentials and mark the user as logged in or logged out.*
- MSR.002: *WeLight must be able to retrieve street lighting data at any time when required, and this data should be the latest information stored in the databases.*
- MSR.003: *As databases might be hosted in a third-party cloud provider, the system must be able to guarantee that this data has not been modified by any third parties.*

- MSR.004: *Intruders may try to impersonate an administrator of the system in order to cause a wide system blackout. The system must be able to restrict access for certain operations according to user´s profile.*
- MSR.005: *The system must be able to guarantee that the metrics collected from street lights are trustworthy (avoid malicious users injecting false data into the system).*

The following table summarizes the security properties identified in D6.1:

Property	Description	Rationale
SP.20	Integrity of *Client input data* is maintained during their lifecycle in the cloud system	Input data are those acquired from sources external to the cloud system, which are used to produce reports and to generate alerts which, in turn, are used to take decisions on public lighting.
SP.21	Integrity of *Client configuration data* is maintained during their life cycle in the cloud system	Client configuration data are acquired from Clients to control both reports and alerts production and alerts distribution.
SP.22	Client configuration data are available to Clients.	Client configuration data are acquired from Clients to control both reports and alerts production and alerts distribution.
SP.23	Integrity of *Client reports* is maintained during their life cycle in the cloud system.	Reports are used to take decisions on public lighting.
SP.24	Client reports are available to Clients.	Reports are used by a given Client to take decisions on public lighting.
SP.25	Integrity of *Client alerts* is maintained during their life cycle in the cloud system.	Alerts are used to take decisions on public lighting.
SP.26	Client alerts are available to Clients according to the corresponding Client configuration data.	Alerts are used by a given Client to take decisions on public lighting.
SP.27	Non-repudiability of origin of Client reports is assured during their life cycle in the cloud system.	Support to resolution of disputes about the fact that a given. Client report has been originated in the cloud system.
SP.28	Non-repudiability of Client alerts is assured during their life cycle in the cloud system.	Support to resolution of disputes about the fact that a given Client alert has been originated in the cloud system.
SP.29	Client accountability with respect to Client configuration data is assured during the life cycle of Client configuration data in the cloud system	Responsibility separation between the entity responsible for the cloud system operation and clients with respect to client alerts and client reports.
SP.30	Integrity of *LCU configuration data* is maintained during their life cycle in the cloud system.	LCU configuration data are used to control LCU.
SP.31	Admin accountability with respect to LCU configuration data is assured during the life cycle of LCU configuration data in the cloud system	Responsibility separation between the entity responsible for the cloud system operation and Admins with respect to client alerts and client reports.
SP.32	Integrity of stored user data is maintained during their life cycle in the cloud system.	Supports the integrity of data managed byWeLight service.
SP.33	Stored user data are available to users.	Supports the availability of data managed by WeLight service.

SP.34	Confidentiality of stored user data is maintained during their life cycle in the cloud system.	WeLight requirement.

Table 1. Security properties identified in D6.1

The security mechanisms (SM) that have been implemented in the context of CUMULUS are listed below:

- SM.1: An encryption mechanism has been added to protect the sensitive information related to each LCU (IP address and password) on the database.
- SM.2: The product has been protected against SQL injection attacks.
- SM.3: TPM is now being used to encrypt the files in the LCU
- SM.4: Integrity of the information about electrical consumption has been implemented. It is checked that the information comes from the correct LCU, that has not been modified in transit and that does not contain errors. This includes checking the freshness of data, this is, that the timestamp is not older than the interval established for capturing data.
- SM.5: The code running on the LCU is signed and compared to the one stored on the server to ensure that no one changed the code running.
- SM.6: A register of deleted data has been implemented to provide persistence of the information as a security measure.
- SM.7: Check of the availability of the server and the time the LCU is up and running.
- SM.8: Login and password for client and admin authentication
- SM.9: Profiling of users in order to restrict access to certain operations. This is done, for instance, to prevent an intruder from feeding the system with false data.
- SM.10: VPN for secure communications with LCU
- SM.11: WeLight server has been configured to redirect any HTTP request to the HTTPS port, therefore forcing to use SSL to ensure the security of communications.

The following table shows how these security mechanisms manage to fulfill the MSR´s:

	SM.1	**SM.2**	**SM.3**	**SM.4**	**SM.5**	**SM.6**	**SM.7**	**SM.8**	**SM.9**	**SM.10**	**SM.11**
MSR.001								X			
MSR.002				X		X	X				
MSR.003	X	X								X	X
MSR.004									X		
MSR.005	X	X	X	X	X					X	X

Table 2. Fulfilment of MSR´s by SM´s

It can be also checked how each SM can be mapped to one or more SP´s:

	SM.1	**SM.2**	**SM.3**	**SM.4**	**SM.5**	**SM.6**	**SM.7**	**SM.8**	**SM.9**	**SM.10**	**SM.11**
SP.20		X		X						X	X
SP.21		X									
SP.22							X				
SP.23		X		X							
SP.24				X		X	X				
SP.25		X									
SP.26						X	X				
SP.27											
SP.28											
SP.29		X									
SP.30	X	X	X		X						
SP.31	X	X	X		X						
SP.32		X		X							
SP.33	X					X	X				
SP.34	X			X							

Table 3. Mapping of SP´s by SM´s

As it can be seen, all the SP´s are addressed by at least one SM except for two of them, those related to non-repudiation. Nevertheless, this kind of property is under research as it is detailed on section 3.2.8. On the other hand SM´s regarding authentication and profiling of users does not address any specific SP. However they address specific MSR´s.

2.3.1. Using TPM to encrypt files

It is worthwhile to go into details about the usage of TPM to encrypt files in the LCU, since it is one of the most relevant mechanisms implemented in the framework of CUMULUS.

Support for the Trusted Platform Module (TPM) has been available in enterprise Linux distributions since SLES 11 [10] and RHEL5.3 [12]. TPM is implemented based on Trusted Computing Group (TCG) specification and one of its many useful applications is to handle dm-crypt passphrases. When using encrypted partitions, one must typically enter one or more passphrases during the boot sequence to allow the kernel to decrypt them. While this is perhaps a desirable characteristic for laptops, it is an impediment to automation in the server environment. TPM can be used in this environment to wrap the passphrases and provide them automatically to the **cryptsetup** command.

Automatic logins can be realized by saving the password as a file and then reading that file when authorization is needed. To make sure this file is not compromised, the best practice is to combine cryptography and directory access control (DAC) by encrypting the password file and setting the proper authorization to it. However most automatic login software expects a plain password file. One can potentially automate the decryption of the password file when that file is needed during automatic logins, but then one will need yet another encrypted password file to decrypt the previous password file. Trusted Platform Module (TPM) provides a clean solution for this recursive problem.

This solution makes use of a set of Platform Configuration Registers (PCRs) that can only be written by the **TPM_extend** operation. The **TPM_extend** operation makes the new PCR a hash of the concatenation of the current value with the new hash that is provided. By design, assigning an arbitrary value to an PCR isn't allowed and makes this **TPM_extend** operation very unique. This design makes key sealing possible.

In a key sealing scenario, the PCR can store a signature of the data that is being extended. The key is sealed (encrypted by an internal specific non-migratable TPM key) and by tying it to a particular PCR value in a way that the key can only be retrieved later from the same TPM only if the PCR value is the same of that at the time of sealing. A key can also be sealed to more than one PCR.

In this particular case, a key to five PCRs (MBR information, bootloader, boot command line, or the kernel image) will be sealed. If any of these PCRs/parameters changes, mounting of the encrypted partition will not be possible. This feature prevents anyone from mounting the partition to other installed operating systems other than the one the partition was originally mounted to, making rootkit impossible. Note that if it is necessary to change any of these five PCR values, planning for migration will be necessary.

The following table can be used as a reference about the files and directories used in the process.

Description	**File / Directory used in the process**
File to temporarily hold the key	/home/temp_plain_key
File of the sealed key	/home/sealed_key
Directory used to mount loopback device	/home/secret_dir
Mapper device of the secured partition Note: This device is always created in the mapper device directory	/dev/mapper/secret
Directory mapped to the above device, where secured files will be made available in plain form	/home/plain

Table 4. Files and directories used in the process of encrypting files by means of TPM [11]

First of all, it is necessary to check if the hardware available is the right one. It is about determining if the hardware is the appropriate to use TPM and find out which TPM version is present in the hardware. The procedure to do so is available in [11].

Subsequently, the Trusted Computing software is installed and configured. The TCG has standardized a software stack that acts above the TPM chip and includes the TPM device driver and TSS. TSS is a Trusted Computing API that provides applications access to the TPM trusted computing functions.

In [11] it is explained how to install an open source implementation of such API, named **TrouSerS**, together with a group of userland tools that use this interface to implement various Trusted Computing Solutions, including the one being accomplished.

The following table includes four software programs that have to be installed in this section and where they should be installed from. *Distro* is defined as the software that comes with the distribution whereas source means that it is necessary to download the software from sourceforge.net and built it from the source.

	SLES11+	**RHEL5.3+**
Software	**Install Software From**	
TrouSerS	distro	distro
tpm_tools v1.3.4+	source	source
trustedgrub	distro	source
cryptsetup	distro	distro

Table 5. Installing software locations [11]

The next step is to create a key (**/home/sealed_key**) that will be used to open the loopback dm-crypt partition. This key will then be sealed to five different PCR´s that were extended to MBR information, bootloader stage part1, bootloader stage part2, boot command line and the kernel image. Once a key is sealed to a PCR, TPM will only allow the key to be retrieved if the content of the PCR remained the same as it was at the moment of the key sealing. Because the key will be sealed to five different PCRs, anyone who attempts to boot the partition/machine from a different installed operating system will not succeed as the content of these PCRs will be different.

Care should be taken because if one of the parameters the key was sealed to (MBR information, bootloader, boot command line, or the kernel image) changes, one will be unable to unseal the key and all encrypted file in the dm-crypted directory will be inaccessible.

It should be highlighted that a random key will be created and saved temporarily to **/home/temp_plain_key**. From it the key will be sealed to the five PCR´s discussed above and it will result in the creation of **/home/sealed_key**. This key will be used afterwards to set up the lookback dm-crypt partition. The particular steps to generate a Trusted State sealed key are detailed in [11].

Next, the dm-crypt lookback partition is set up. Concretely, an available loopback device **(/dev/loop0**) is initiated by associating it to a directory (**/home/secret_dir**). Then the device will be set up to be a LUKS-encrypted partition using the sealed key (**/home/sealed_key**) created previously. This partition will then be mapped to a mapper device (**/dev/mapper/secret**) and mounted at a plain directory (**/home/plain**) for use. The steps to do so are explained in [11].

After having set up the dm-crypt loopback partition, the sensitive files will be moved to this partition and the original file will be replaced to a symbolic link to the new location. This practice centralizes the secured files and removes the need to encrypt each file separately. Note that if the plain partition is unmounted and any parameter sealed in the key (MBR information, bootloader, boot command line, or the kernel image) changes, the symbolic link will not work and the files will not be accessible.

In [11] a shell script is provided to automate moving a particular file to the decrypted loopback partition (**/home/plain**). In addition, the shell script replaces the original file location with a symbolic link to the loopback path. It is necessary to edit this script if the secured partition does not reside at **/home/plain.**

Later on, some steps described in [11] have to be followed in order to make this change persistent. The steps are different depending on the Linux distribution one is working with: SLES11 or RHEL5.4. Once this is done, the dm-crypt loopback partition is set up to persist across a reboot. Next the script that was previously created can be used to move the sensitive files to the secured partition. Bear in mind that any changes to the five parameters corresponding to the PCR´s will cause the sealed key to become invalid. Extreme care must be taken to avoid the situation where data is unrecoverable.

Finally, it will be verified whether the secured partition is really secured. The instructions to do this are provided in [11].

3. Certification of security properties

This section will go into the work carried out to certify security properties in the Smart City scenario by using the CUMULUS Framework in depth. This section will have two subsections: one to explain the envisaged work, describing at a functional level the security properties that are to be certified, and that are specified in [8] and [9]. The other one will be devoted to deal with those security properties that are under research and could also be certified. In some cases, the work to be accomplished is not totally defined since several issues are still open. All the information regarding the ongoing research will be reported so as to reflect the work done so far. It is necessary to highlight that, as the DoW explains, the ongoing work will be reported in an internal deliverable due February 2015, a brief summary will be provided in D6.5 due January 2015, and finally a detailed explanation of all the activities and achievements regarding certification of security properties will be included in D6.6, due September 2015.

3.1. Work planned

As mentioned above, the security properties to be certified are specified in [8] and [9]. The following table summarizes these properties:

	Property and its attributes	**Cloud Stack (CS) and Target of Certification (ToC)**	**Approach**	**Technique**

1	AIS:confidentiality:external-data-exchange-confidentiality
	Performance attribute: verified (Boolean) = true
		CS = PaaS
		ToC = VPN
			Verify that the data is transmitted through the VPN	Monitoring

2	AIS:authentication:network-authenticated-server-access
	Performance attribute: verified (Boolean) = true
		CS = SaaS
		ToC = application server
			Verify HTTPS communication	Monitoring

3	IVS:isolation:tenant-isolation-level
	Performance attribute: Level (int) = 2
		CS = Infrastructure
		ToC = Virtual Machine /
			Verify relation between: 1) tenants and infrastructure, 2) tenants and users´authentication	Testing

		OpenStack		

4	AIS:confidentiality:external-data-exchange-confidentiality
	Performance attribute: verified (Boolean) = true
		CS = Service
		ToC = login Web Interface
			Verify external (Service) channel encryption	Testing

	AIS:confidentiality:external-data-exchange-confidentiality
	Performance attribute: verified (Boolean) = true
		CS = Infrastructure
		ToC = OpenStack API
			Verify internal infrastructure channel encryption	Testing

5	AIS:confidentiality:service-provider-data-access-level
	Performance attribute: Level (int) = 2
		CS = Service + Platform + Infrastructure (multilayer)
		ToC = Web Interface + MySQL DB + File System + OpenStack
			Verify 1) (Service Level) Authorization 2) DB grants, 3) File System Access Policy, 4) Tenant isolation, 5) Internal infrastructure channel encryption	Testing

6	DSI:data-leakage-control:data-leakage-prevention
	Performance attribute: pattern (string) = Credit Card string
		CS = Service + Platform + Infrastructure (multilayer)
		ToC = Web Interface + MySQL DB + File System + OpenStack
			Check 1) HTTPS service 2) Verification of DB encryption 3) File System Access Policy 4) Verification of use of an encrypted channel (Infrastructure)	Testing

Table 6. Security properties defined for Smart Cities pilot

The figures below show the infrastructure used to certify the security properties. The dashboard is the graphical user interface from which the certification processes (both monitoring and testing) are launched. The dashboard communicates with the “Glue” Component, which is composed of the Monitoring and the Testing Manager.

On the one hand, the Monitoring Manager receives the events captured by the Event Captor and sent by means of the Open Fire bus, as it is explained in sections 3.1.1 and 3.1.2.

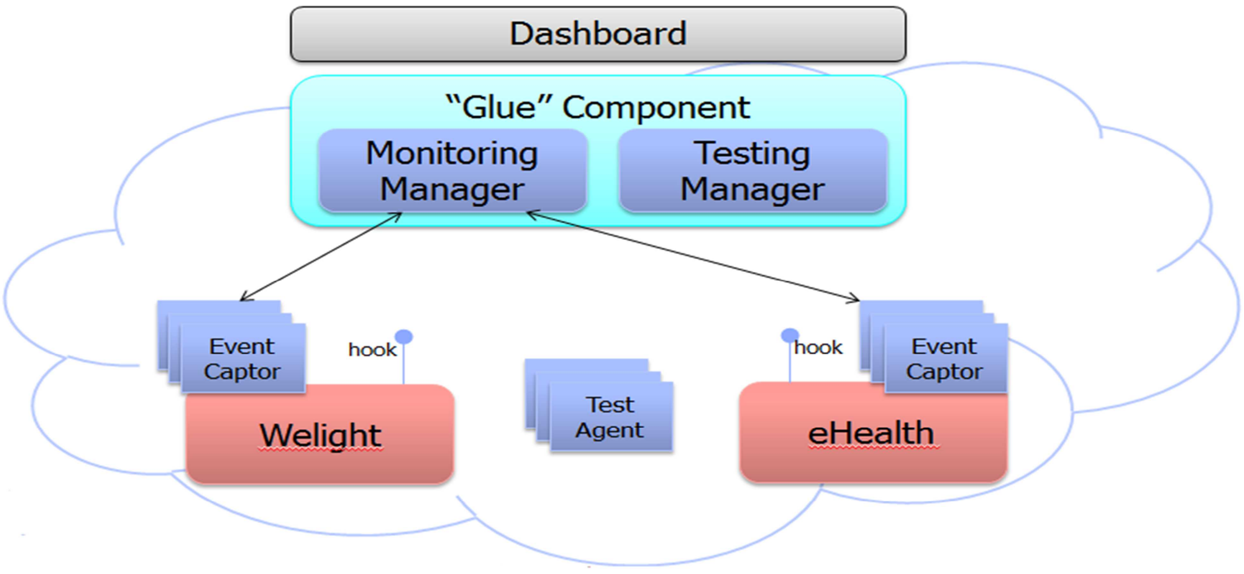

Figure 2. Monitoring infrastructure

On the other hand, the Testing Manager receives the evidence from the Test Agent, which in turn collects such evidence by using hooks provided both in the Smart Cities and in the eHealth pilot.

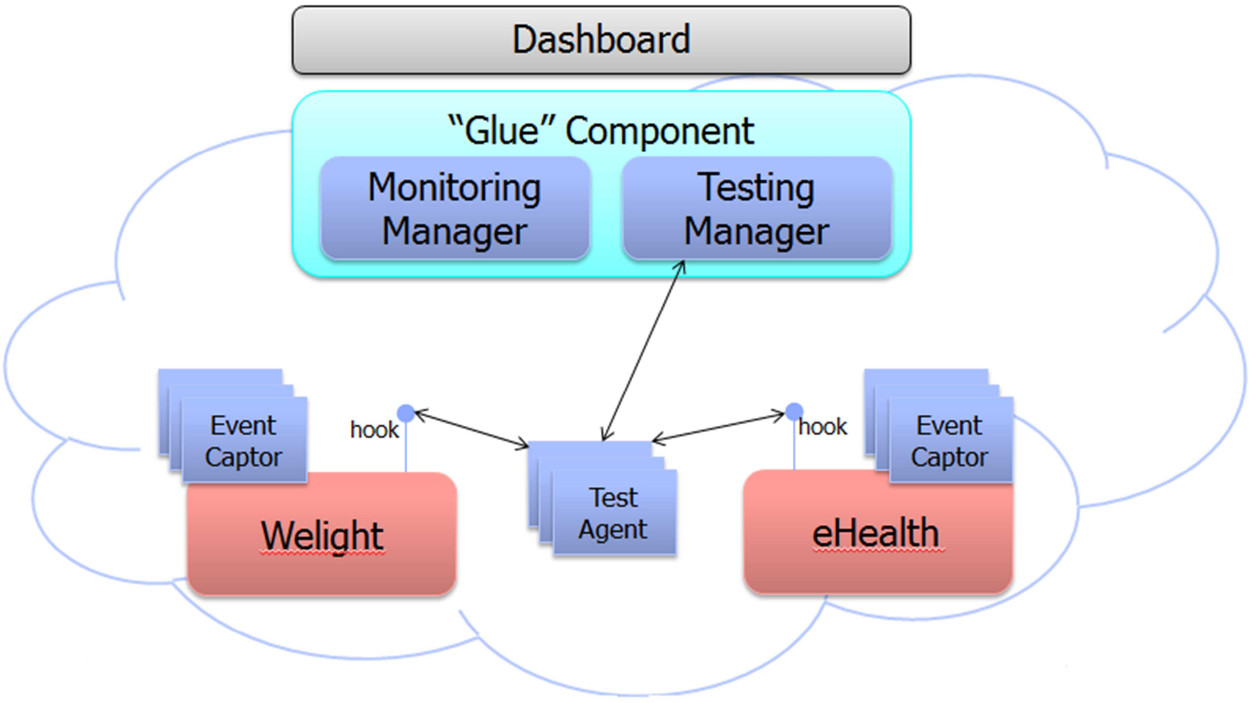

Figure 3. Testing infrastructure

3.1.1. External data exchange confidentiality

This property has been addressed for three different ToCs: the VPN, the login Web Interface and the OpenStack API

VPN

This property implies the ToC offering a confidential network channel, for data exchanges with external parties. Since the ToC does not include the external parties, this property alone does not imply authentication of the external parties nor does it imply that these external parties are able to maintain the confidentiality of the data.

It is necessary to ensure that the VPN established between WeLight Virtual Machine and the LCU´s is working properly and the communications between both sides are protected. Otherwise there could be a suspicion of a Man-In-The-Middle attack tacking place.

The certification of the correct operation of the VPN will be accomplished by means of monitoring techniques.

The idea is to place an agent on WeLight VM side. This agent will:

- Monitor the sending of packets to the LCU. The information that will be of interest is the IP address of the packet recipient. This should be the VPN IP address.
- Monitor the arrival of packets to the WeLight server from the LCU. These packets will be extracted their origin IP address. This should coincide with the VPN IP address.

This agent is the Event Captor. When it detects the sending/arrival of a packet, it extracts the destination/origin IP address and parses this information along with other relevant data (such as the entity name –Sender, Receiver, Event Captor… - the port or the timestamp, for instance) to an XML template. This file will be sent to the Monitor Manager. The method established to send this packet will be event generation. An OpenFire event bus will be used. When working with events it is very common to consider an approach publisher/subscriber. The publisher is the entity interested in sending the event and the subscriber is the one which will use the information included in the event.

For further information see [6].

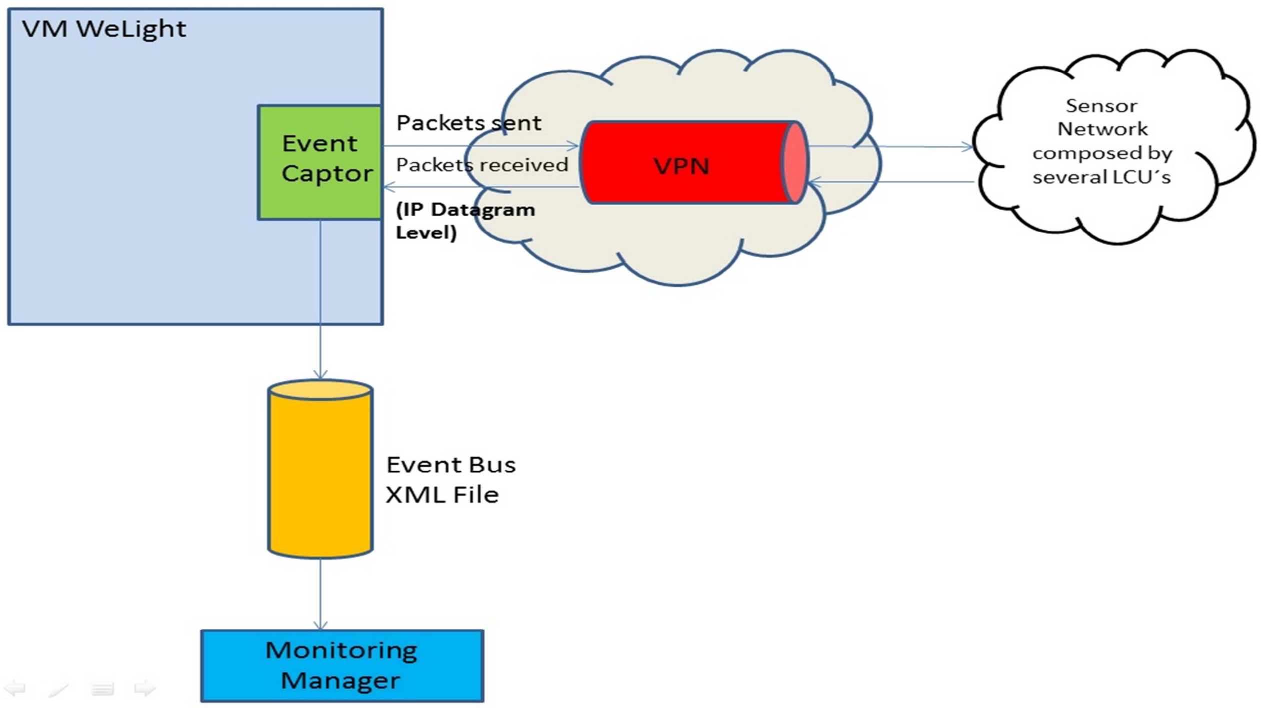

Figure 4. Event caption and sending through the Event Bus

Each time there is an arrival/sending of a packet, some data are extracted and converted to a text string with a XML format. The following characteristics can be retrieved in this XML string:

- The IP of the Event Captor, which is for the moment the IP of the VM (192.168.43.26) and the port of the exchanged message, which is for the moment the UDP/TCP port (10022)
- The IP and the port of the Receiver entity, which correspond to the final destination of the packet.
  - For the sending, the IP is unknown because the VM only knows the IP of the VPN tunnel, not the one of the LCU.
  - For the receiving, the IP and the port are the same than the ones of the Event Captor (192.168.43.26:10022)
- The IP and the port of the Sender entity.
  - For the sending, the IP and the port are the same than the ones of the Event Captor (192.168.43.26:10022)
  - For the receiving, the IP is unknown because the LCU only knows the IP of the VPN tunnel, not the one of the VM.
- The IP and the port of the destination, to which is sent the packet. For both sending and receiving, this corresponds to the IP of the VPN tunnel and its port.
  - For the sending, the value is actually 10.9.8.1:10022
  - For the receiving, the value is actually 10.9.8.10:10022
- The ID of the exchange. Indeed, each time the VM sends a request, it expects a response. This pair request-response may have the same ID, and two different pair can't have the same ID.
- The name and the status of the operation
  - For the sending, the name is LCUCall and the status is REQ-B
  - For the receiving, the name is LCUCall and the status is RES-A
- The type of the event
  - For the sending, the type is ServiceOperationCallStartEvent
  - For the receiving, the type is ServiceOperationCallEndEvent

The created XML´s are shown below:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Event xmlns="http://www.slaatsoi.org/eventschema">
<EventID>
    <ID>21</ID>
    <EventTypeID>ServiceOperationCallEndEvent</EventTypeID>
</EventID>
```

```
<EventContext>
    <Time>
             <Timestamp>1414425598519</Timestamp>
            <CollectionTime>2014-10-27T16:59:58</CollectionTime>
            <ReportTime>2014-10-27T16:59:58</ReportTime>
     </Time>
    <Notifier>
            <Name>Malaga Server</Name>
             <IP>192.168.43.26</IP>
            <Port>10022</Port>
    </Notifier>
    <Source>
            <SwServiceLayerSource>
                    <Sender>
                            <name>ucapi0010</name>
                            <ip>0.0.0.0</ip>
                             <port>10022</port>
                            <serviceEPR />
                    </Sender>
                    <Receiver>
                            <name>Malaga Server</name>
                            <ip>192.168.43.26</ip>
                             <port>10022</port>
                            <serviceEPR />
                    </Receiver>
            </SwServiceLayerSource>
     </Source>
 </EventContext>
 <EventPayload>
    <InteractionEvent>
            <OperationName>LCUCall</OperationName>
            <OperationID>21</OperationID>
            <Status>RES-A</Status>
            <Parameters>
                    <Argument>
                            <Simple>
                                    <ArgName>packetSource</ArgName>
                                    <ArgType>string</ArgType>
                                    <Direction>OUT</Direction>
                                    <Value>10.9.8.10:10022</Value>
                             </Simple>
                      </Argument>
               </Parameters>
        </InteractionEvent>
     </EventPayload>
</Event>
```

Figure 5. Event generated when receiving a packet

```
  <?xml version="1.0" encoding="UTF-8" ?>
  <Event xmlns="http://www.slaatsoi.org/eventschema">
  <EventID>
      <ID>21</ID>
      <EventTypeID>ServiceOperationCallStartEvent</EventTypeID>
  </EventID>
  <EventContext>
      <Time>
             <Timestamp>1414425594930</Timestamp>
             <CollectionTime>2014-10-27T16:59:54</CollectionTime>
             <ReportTime>2014-10-27T16:59:54</ReportTime>
      </Time>
  <Notifier>
      <Name>Malaga Server</Name>
      <IP>192.168.43.26</IP>
      <Port>10022</Port>
  </Notifier>
  <Source>
      <SwServiceLayerSource>
             <Sender>
                    <name>Malaga Server</name>
                    <ip>192.168.43.26</ip>
                    <port>10022</port>
                    <serviceEPR />
             </Sender>
```

```
            <Receiver>
                  <name>ucapi0010</name>
                  <ip>0.0.0.0</ip>
                  <port>10022</port>
                  <serviceEPR />
             </Receiver>
      </SwServiceLayerSource>
   </Source>
</EventContext>
<EventPayload>
  <InteractionEvent>
      <OperationName>LCUCall</OperationName>
      <OperationID>21</OperationID>
      <Status>REQ-B</Status>
      <Parameters>
            <Argument>
                  <Simple>
                        <ArgName>packetDest</ArgName>
                        <ArgType>string</ArgType>
                        <Direction>IN</Direction>
                         <Value>10.9.8.1:10022</Value>
                  </Simple>
            </Argument>
      </Parameters>
   </InteractionEvent>
 </EventPayload>
</Event>
```

Figure 6. Event generated when sending a packet

Once this String is created, it is sent to the Event Bus through a protocol XMPP. This protocol allows a system of publisher/subscriber over a channel. Indeed, when a publisher publishes a message on a channel, all the subscribers will receive it. For the project, the name of the channel is "EverestEventChannel", and one publisher is the VM. So, each time a packet is received or sent, the XML String will be published on the channel.

The figure below depicts a summary of the main steps.

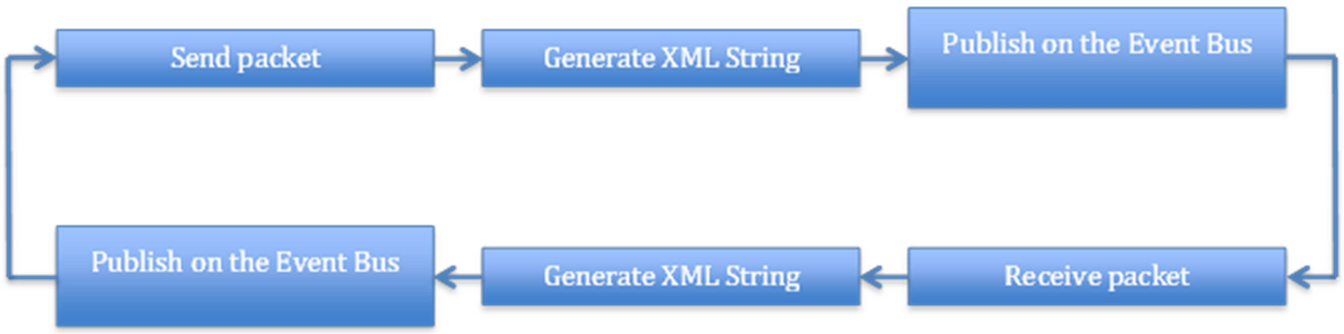

Figure 7. Main steps of the monitoring process for certifying the correct operation of the VPN

The Monitor Manager subscribes to the Event Bus in order to receive the events. The events are then matched by the Monitor Manager with the rule for the considered security property, which the interested reader may find in D3.2. The outcome of such rule is used in order to release and constantly update the status of the corresponding certificate.

Three direct benefits of certifying this security property have been identified:

- The information between the Virtual Machine and the LCU is transmitted through the network in a secure manner thanks to the VPN and in consequence the client can feel reassured that no Man-In-The-Middle attack will take place, and this will improve the appealing of WeLight.
- The operation is improved as it is possible to have a pool of static IP addresses that are reserved and allocated to the LCU´s.
- Finally, no third party (malicious or not) can interact with the devices, which brings improved security and better bandwith control.

Login Web Interface

Property **AIS:confidentiality:external-data-exchange-confidentiality** guarantees that the ToC (the Login Web Interface in this case) offers a confidential network channel for data exchange with external parties. In Welight scenarios this property reflects the necessity of an encrypted channel between the web service itself and the users.

The test aims to certify that the channel provided by the Welight service is encrypted and the information is exchanged through a TSL/SSL communication. The test, first checks if the SSL/TSL support is enabled, then it checks the server certificate validity and the robustness of all supported ciphers.

The log analyzed from our probe is shown below and important parts are highlighted and in bold.

```
    Starting Nmap 6.40 ( http://nmap.org ) at 2014-11-06 11:31 UTC
    Nmap scan report for 192.168.43.62
    Host is up (0.00092s latency).
    PORT     STATE SERVICE
    8080/tcp open  http-proxy
    |          ssl-cert:          Subject:          commonName=cumulus-project.sytes.net/organizationName=Wellness
Te\xC3\x83lecom/stateOrProvinceName=Seville/countryName=ES
    |                   Issuer:                     commonName=cumulus-project.sytes.net/organizationName=Wellness
Te\xC3\x83lecom/stateOrProvinceName=Seville/countryName=ES
    | Public Key type: rsa
    | Public Key bits: 2048
    | Not valid before: 2014-06-16T14:57:24+00:00
    | Not valid after:  2015-06-16T14:57:24+00:00
    | MD5:   d736 7f12 72f9 89df 9200 c32e 75cf cf22
    |_SHA-1: c8c5 ceb2 5086 6013 f7a6 50fd 22db 2535 7efc 6d38
    | ssl-enum-ciphers:
    |   SSLv3:
    |     ciphers:
    |       TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA - strong
    |       TLS_DHE_RSA_WITH_AES_128_CBC_SHA - strong
    |       TLS_DHE_RSA_WITH_AES_256_CBC_SHA - strong
    |       TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA - strong
    |       TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA - strong
    |       TLS_DHE_RSA_WITH_SEED_CBC_SHA - strong
    |       TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA - strong
    |       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA - strong
    |       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA - strong
    |       TLS_ECDHE_RSA_WITH_RC4_128_SHA - strong
    |       TLS_RSA_WITH_3DES_EDE_CBC_SHA - strong
    |       TLS_RSA_WITH_AES_128_CBC_SHA - strong
    |       TLS_RSA_WITH_AES_256_CBC_SHA - strong
    |       TLS_RSA_WITH_CAMELLIA_128_CBC_SHA - strong
    |       TLS_RSA_WITH_CAMELLIA_256_CBC_SHA - strong
    |       TLS_RSA_WITH_RC4_128_SHA - strong
    |       TLS_RSA_WITH_SEED_CBC_SHA - strong
    |     compressors:
    |       NULL
    |   TLSv1.0:
    |     ciphers:
    |       TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA - strong
    |       TLS_DHE_RSA_WITH_AES_128_CBC_SHA - strong
    |       TLS_DHE_RSA_WITH_AES_256_CBC_SHA - strong
    |       TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA - strong
    |       TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA - strong
    |       TLS_DHE_RSA_WITH_SEED_CBC_SHA - strong
    |       TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA - strong
    |       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA - strong
    |       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA - strong
    |       TLS_ECDHE_RSA_WITH_RC4_128_SHA - strong
    |       TLS_RSA_WITH_3DES_EDE_CBC_SHA - strong
    |       TLS_RSA_WITH_AES_128_CBC_SHA - strong
    |       TLS_RSA_WITH_AES_256_CBC_SHA - strong
    |       TLS_RSA_WITH_CAMELLIA_128_CBC_SHA - strong
    |       TLS_RSA_WITH_CAMELLIA_256_CBC_SHA - strong
    |       TLS_RSA_WITH_RC4_128_SHA - strong
    |       TLS_RSA_WITH_SEED_CBC_SHA - strong
    |     compressors:
    |       NULL
    |   TLSv1.1:
    |     ciphers:
    |       TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA - strong
    |       TLS_DHE_RSA_WITH_AES_128_CBC_SHA - strong
    |       TLS_DHE_RSA_WITH_AES_256_CBC_SHA - strong
    |       TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA - strong
    |       TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA - strong
    |       TLS_DHE_RSA_WITH_SEED_CBC_SHA - strong
    |       TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA - strong
    |       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA - strong
    |       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA - strong
    |       TLS_ECDHE_RSA_WITH_RC4_128_SHA - strong
    |       TLS_RSA_WITH_3DES_EDE_CBC_SHA - strong
    |       TLS_RSA_WITH_AES_128_CBC_SHA - strong
    |       TLS_RSA_WITH_AES_256_CBC_SHA - strong
    |       TLS_RSA_WITH_CAMELLIA_128_CBC_SHA - strong
```

```
    |     TLS_RSA_WITH_CAMELLIA_256_CBC_SHA - strong
    |     TLS_RSA_WITH_RC4_128_SHA - strong
    |     TLS_RSA_WITH_SEED_CBC_SHA - strong
    |   compressors:
    |     NULL
    |  TLSv1.2:
    |   ciphers:
    |     TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA - strong
    |     TLS_DHE_RSA_WITH_AES_128_CBC_SHA - strong
    |     TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 - strong
    |     TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 - strong
    |     TLS_DHE_RSA_WITH_AES_256_CBC_SHA - strong
    |     TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 - strong
    |     TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 - strong
    |     TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA - strong
    |     TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA - strong
    |     TLS_DHE_RSA_WITH_SEED_CBC_SHA - strong
    |     TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA - strong
    |     TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA - strong
    |     TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 - strong
    |     TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 - strong
    |     TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA - strong
    |     TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 - strong
    |     TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 - strong
    |     TLS_ECDHE_RSA_WITH_RC4_128_SHA - strong
    |     TLS_RSA_WITH_3DES_EDE_CBC_SHA - strong
    |     TLS_RSA_WITH_AES_128_CBC_SHA - strong
    |     TLS_RSA_WITH_AES_128_CBC_SHA256 - strong
    |     TLS_RSA_WITH_AES_128_GCM_SHA256 - strong
    |     TLS_RSA_WITH_AES_256_CBC_SHA - strong
    |     TLS_RSA_WITH_AES_256_CBC_SHA256 - strong
    |     TLS_RSA_WITH_AES_256_GCM_SHA384 - strong
    |     TLS_RSA_WITH_CAMELLIA_128_CBC_SHA - strong
    |     TLS_RSA_WITH_CAMELLIA_256_CBC_SHA - strong
    |     TLS_RSA_WITH_RC4_128_SHA - strong
    |     TLS_RSA_WITH_SEED_CBC_SHA - strong
    |   compressors:
    |     NULL
 |_  least strength: strong
```

Figure 8. NMAP results. Relevant information is highlighted

3.1.2. Network authenticated server access

The ToC (in this case the application server) provides a communication channel between a client and itself, which offers the following guarantees:

1) The client receives assurance that there is effective communication with the ToC
2) The integrity of the exchange of data between the client and the ToC is protected (i.e with guarantees that exchanged messages are both authentic and fresh).

This property does not cover the confidentiality of the data exchanged.

The property could be considered as 'verified' if the client communicates with the ToC using SSL/TLS and verifies ToC´s authenticity with a certificate. The system must verify, using SSL/TLS, the authenticity of the user and the webservice. All the insecure communications (port 80) must be redirected to the HTTPS port (443), so as to be considered secure traffic.

The methodology followed is the same that was presented in section 3.1.1. The evidences will be obtained by analyzing an XML file transmitted by means of the OpenFire event bus, whose information is obtained from a captured HTTP request. What changes in this case is twofold:

1. In the case of VPN certification the evidences were obtained from IP datagrams, this is, the work is done at the IP level. In the case of secure communications certification the evidences are obtained from HTTP requests and responses, this is, the work is done at the HTTPS level.
2. On the other hand, the kind of information that is required to be published in the event bus. While just the IP address was needed as evidence in the case of certifying the correct operation of the VPN, here the following information is required:
   2.1. For each request received on the HTTP port an event to be sent is needed. The required information is the kind of request that was received: a secure one (using SSL/TLS) or an ordinary one.
   2.2. For each response on the HTTP port, the information will be the status code of the response and the location header field, if provided, or empty string otherwise.
   2.3. For each request received on the HTTPS port an event to be sent is needed. The required information is the same as in the case 2.1.
   2.4. For each response on the HTTPS port, the information will be the same as in the case 2.2

For further information see [7].

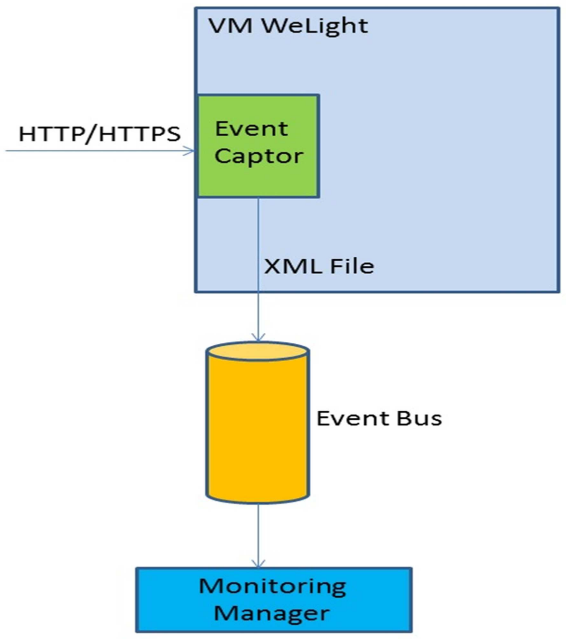

Figure 9. Secure Communications

3.1.3. Tenant Isolation Level

This property guarantees the isolation of data between tenants according to a performance attribute defined as level [15]:

- Level 0 means no strict isolation is provided between tenants
- Level 1 means cross-tenant reading or writing to active resource data is not possible
- Level 2 means that level 1 is supported, with the addition that when two tenants share the same hardware, cross tenant reading any data is impossible.

In these testing activities, hooks are not strictly needed, except for the fact that an administrative privilege for inspecting tenants and user privileges on file system is needed. The probes act as an administrator accessing hypervisor, file system and ACL. More in details, it is the absence of injection relation between tenants and user, and the non-overlapping of user´s privileges on what is tested.

The steps followed to accomplish the test are described below:

1) Check the number of tenants in the Cloud Environment and in the VM: if one only tenant is present the property is automatically demonstrated at level 2.
2) If test at step 1 is not true, the property could be still certifiable at level 1 by testing relations between users and the hypervisor´s tenants.
3) If the previous test case at step 2 is successful (level 1 certified), level 2 can be achieved only with additional testing on the hypervisor used. In particular, it has to do with hypervisor isolation checking.

For further information, this is widely detailed in [15].

3.1.4. Confidentiality Data Access Level

This property matches **AIS:confidentiality:data-access-level**. It describes the level of data confidentiality in the ToC with respect to the personnel operating the ToC. The property is expressed in terms of performance objective, independently of the underlying mechanisms, with a level (as performance attribute) between 0 and 4. Testing has been done at the following levels [16]:

- Level 1: when data is only accessible within the ToC and no external entities have access to the data.
- Level 2: including some further restrictions namely: technical and organizational measures such that data may only be accessible by the ToC personnel for regular operational purposes, and access is subject to individual authentication, authorization and traceability controls.

This property is the object of a multilayer testing process carried out at different levels of the cloud stack including SaaS, PaaS and IaaS. The tests are addressed to the WeLight web service itself and to all services that support the welight provisioning: File System access, database grants, Infrastructure Manager security. To accomplish this process, hooks must be provided at different cloud stack layers. The tests are addressed to the WeLight web service itself and to all services that support the WeLight provisioning: File System access, database grants and Infrastructure Manager security. The certification condition is expressed in Table 6, and here below the single test is described.

To check the File System access policy a test tries to access in read/write/execution mode different files in the WeLight FS. The result should be compliant with a given access policy.

To check Database Grants, a test tries to access in read/write/update different tables into WeLight DB and the result should be compliant with the given access policy.

The privacy of the infrastructure manager is related to tenant isolation and data exchange confidentiality. The tenant isolation is guaranteed by the fact that Wellness is the only owner of the cloud infrastructure, while the data exchange confidentiality, with Openstack as IaaS, is guaranteed by the use of SSL/TSL in its components: nova (deployment service), keystone (identity service) and horizon (the Openstack web interface).

Indeed, all results from all the tests are collected and gathered in order to identify if all the conditions are satisfied.

This process is explained on detail in [16]. The figures below show the logical flow for the verification of both the database verification of grants and the file system verification of access policy.

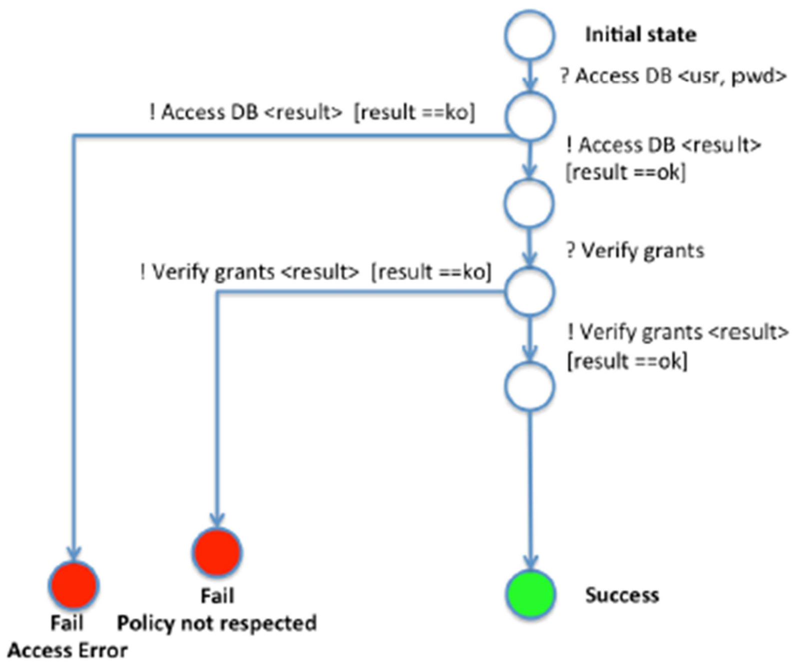
Initial state
? Access DB <usr, pwd>
! Access DB <result> [result ==ko]
! Access DB <result> [result ==ok]
? Verify grants
! Verify grants <result> [result ==ko]
! Verify grants <result> [result ==ok]
Fail Access Error
Fail Policy not respected
Success

Figure 10. Flow for database verification of grants [16]

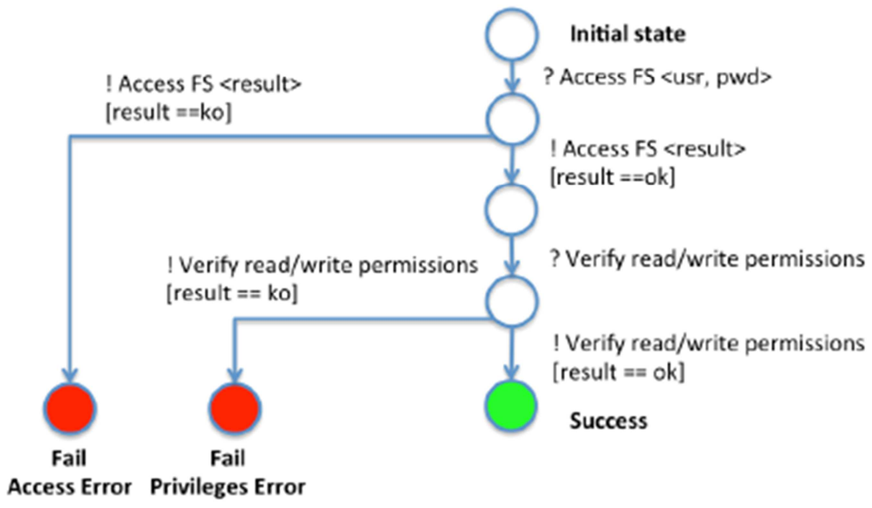
Initial state
? Access FS <usr, pwd>
! Access FS <result> [result ==ko]
! Access FS <result> [result ==ok]
? Verify read/write permissions
! Verify read/write permissions [result == ko]
! Verify read/write permissions [result == ok]
Success
Fail Access Error
Fail Privileges Error

Figure 11. Flow for system verification of access policy [16]

3.1.5. Data Leakage Prevention

The goal is to protect sensitive data for the ToC over different layers of the cloud stack. When data leaving the ToC matches certain patterns or heuristics it is blocked and an alert is generated.

The approach will be to do several tests such as checking the HTTPS service, checking the verification of the database encryption, checking the file system access policy and finally checking the verification of use of an encrypted channel. To do these tests, hooks will be provided.

The figure shows a graphical representation of these tests and the different layers of the cloud stack where they are carried out.

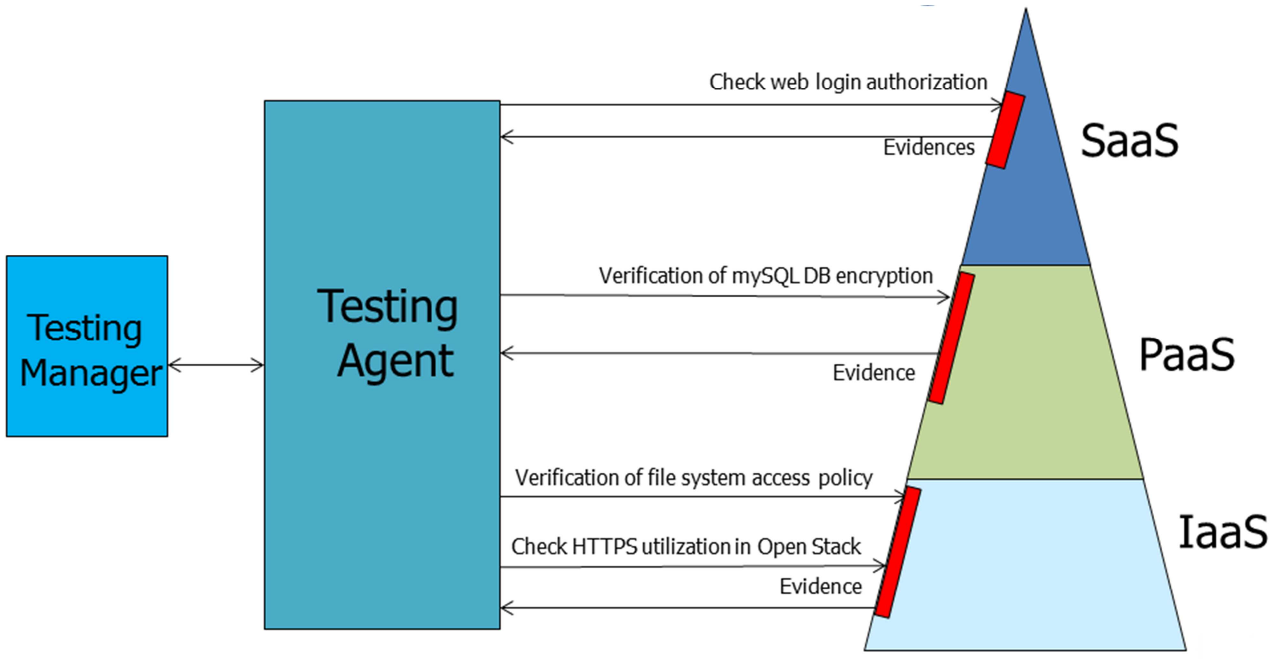
Testing Manager
Testing Agent
Check web login authorization
Evidences
SaaS
Verification of mySQL DB encryption
Evidence
PaaS
Verification of file system access policy
Check HTTPS utilization in Open Stack
Evidence
IaaS

Figure 12. Multilayer Certification. Data Leakage Prevention

HTTPS web server enabled and SST/TSL over Openstack services are tested using NMAP as described in 3.1.4. Each test is addressed to a different service. As depicted in Figure 7, probe will test web service https connection to 192.168.43.68:8080 (where the https web service responds), then the probe will test OpenStack connecting to 192.168.43.70 and address its script to ports 5000, 8774,443 where the services are running (see Figure 9).

To check the DB encryption a dedicated probe was developed. It requires that there is at least one entrance into the DB. The probe accesses as admin in Welight, then it acquires the data (LUC's IP and access information) from the web page. The same data are collected from the database and decrypted using the related key. Once both values (from web page and from DB) are available they are compared in order to check if encryption works (see Figure below).

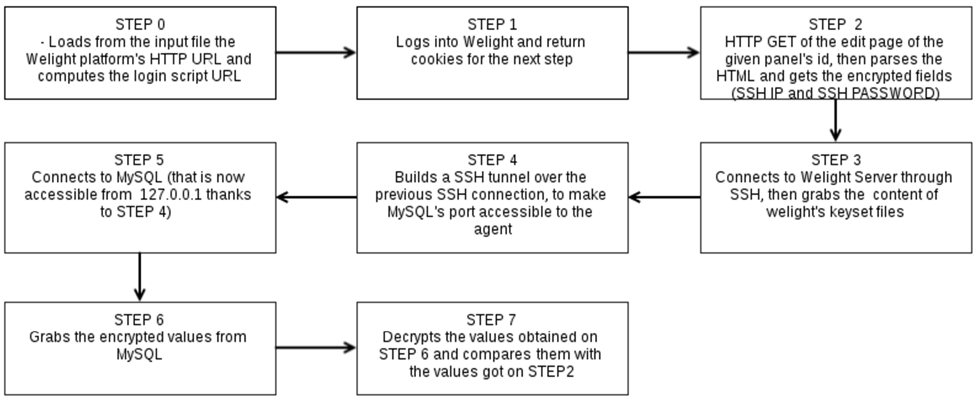
STEP 0
- Loads from the input file the Welight platform's HTTP URL and computes the login script URL
STEP 1
Logs into Welight and return cookies for the next step
STEP 2
HTTP GET of the edit page of the given panel's id, then parses the HTML and gets the encrypted fields (SSH IP and SSH PASSWORD)
STEP 3
Connects to Welight Server through SSH, then grabs the content of welight's keyset files
STEP 4
Builds a SSH tunnel over the previous SSH connection, to make MySQL's port accessible to the agent
STEP 5
Connects to MySQL (that is now accessible from 127.0.0.1 thanks to STEP 4)
STEP 6
Grabs the encrypted values from MySQL
STEP 7
Decrypts the values obtained on STEP 6 and compares them with the values got on STEP2

Figure 13. DB Encryption Test Flow

File System Access policy is tested through a probe that access the Welight machine in ssh. Then it tries to access specific files in read and write mode in order to check the given access policy. The tested files are the DB encryption key and web service file configuration. Both files are owned by the cumulus user and their chmod is set on 600, only the owner can read and write the file (see Figure 11).

The implementation of this multilayer certification increases the appealing of the product and allows to stand out from the competitors. This is very important, since WeLight main market is public lighting and has to compete in tenders.

So far, public administration has not explicitly asked security but market studies done at Wellness Telecom confirm that the trend is changing and security will be a main concern regarding public lighting. In such sense Wellness Telecom is running ahead of the market.

3.2. Other ToCs and properties under research

3.2.1. Data leakage prevention

This security property ensures that the sensitive information stored in the database is properly encrypted.

In this case, the ToC is WeLight database and in particular the table where the IP addresses and the SSH passwords for each LCU are stored. No one breaking the security and accessing the database should be able to obtain this information in plain text.

In order to protect the sensitive information stored in the service database, the application will support symmetric encryption for private fields, such as the IP address of each LCU or its SSH passwords.

An attacker who gets to the database could use these fields to connect to each LCU by SSH (the only active connection way). So to reduce the risk of this kind of attacks, the application will use the Keyzcar cryptographic toolkit [1] that avoid plain reading in the event of the database getting compromised.

The mechanism to ensure sensitive data are encrypted would be a read query to the database. The table below shows the fields that compose any row of the table involved in the process. Both the password and the IP address are shown encrypted.

id	name	latitude	longitude	user	password	ip
1	Cumulus	37401890	-6005724	root	enc_str:::AFHunKAA8t6UeQX2brXLNqR-2yzZcFfFqgLPvJK1lMGzX6OxtmHG9fm9shrCVRRDXaD_Cov9_9Yv	enc_str:::AFHunKB0lI-HlEPRbrIn0JIqQ7y0hvNpeOQGlznA_tTUmrty5hxNYpauxxAYs5clQKH1fYemwxAu

Table 7. Web Server electrical panel

Both static and dynamic tests can be defined in order to accomplish this.

Static test includes the replica of the encryption process from LCU to WeLight Server. A test agent simulates to be an LCU and, using the encryption tool, sends the IP and the password to the database interface to be stored. Then the correctness of the encryption on the database side will be checked.

As for the on-line/dynamic tests the following approaches can be implemented:

1) The IP pattern ('x.x.x.x') can be checked so as to ascertain whether the IP address is stored in plain text, but this process provides weak evidence, because the fact that the pattern does not fit the IP format does not mean that the IP is encrypted.
2) Having access to the database, the stored encrypted passwords could be used to log in, expecting a failure in the login process. The evidence would be still weak, because in the case it fails, it can only be inferred that the passwords do not work as they are stored in the database.
3) The interface used to store the IP address and password can be made available, so that a new password and IP can be added and then check if they are encrypted or not (to check this it is necessary to decrypt the challenge and compare with the sent password and IP). Once this is done, the added entries will be deleted.

The encryption process is transparent to the application so there is no way to know the algorithm being used, the key length, the mode of operation, or the initialization vectors.

In the end, the most effective way to accomplish this certification is to simulate the insertion/modification of a LCU object (by invoking an API, being the parameter in plain text), and checking on the database whether or not the data were protected. This corresponds with option 3.

This will be the first approach to certify this security property. Besides, during year 3 the works will be focused on certifying this property by using TC Proofs, which will support monitoring-based certification model. This will strengthen the certification process. The application used to perform the encryption will play the role of ToC.

It will be necessary to examine what security mechanism the application uses to encrypt data and if it is properly recognized by the community or certified so as to rely on it. Given that, a CA can use TC-based certification to ensure the integrity of the application together with its libraries used to perform encryption, and with ToC description identifying what security mechanism implementations are used.

Under the condition that the monitoring Certification Manager (CM) recognizes the security mechanism implementations (part of ToC) as certified by TC-based CM, it can be deduced that the application (at time of monitoring) is trustworthy to correctly perform data encryption operation (i.e. encrypt LCU IP and password before storing those in a database).

3.2.2. Confidentiality data access level

Storage services must ensure that the information stored in persistent medium remains confidential for all except the users of the service and authorized personnel of the underline service. We are considering three different ToC´s: the file system in the LCU, the settings file and WeLight database driver.

File System in the LCU

So far, LCU themselves have not been used in the context of CUMULUS. They have been simulated by using Raspberry PI. This is a credit-card sized computer that can be used in electronics projects developed by the Raspberry PI Foundation with the intention of promoting the teaching of basic computer science in schools [2].

Each LCU stores its code and data into the user file system. Using a Raspberry PI as a LCU will increase the risk to access to these files due to the ease to read the SD card of this device by a possible hacker.

The main way to protect the confidentiality of these files is to create a fully encrypted partition where the code and data will be located. The use of TPM component will be helpful to implement a secured area inside the files system of the SD card.

The mechanism to check the encrypted partition is well mounted would be through an API call of the web service of the application. The application will verify the mounted partition executing a ‘dmsetup status’ command through a SSH connection and it will return the status of the encrypted partition.

The format of the information returned by it must be analyzed on detail. Depending on what it returns and its format, it will be likely to be monitored by using monitoring tools.

This property matches **AIS:confidentiality:data-access-level**. Administrator users may access to the code and data in order to perform non-regular operations so the level assigned to this property is 3.

Settings file

Given that a settings file contains sensitive information, such as the database password, it is necessary to limit access to it. For example, change its file permissions accordingly. This is especially important in a shared-hosting environment. When working with Linux OS, the most secure permission level is 600 (-rw-------).

The goal here is to certify that the access to the settings file is restricted properly.

Administrator users may access to the code and data in order to perform non-regular operations so the level assigned to this property is 3.

So far, several approaches have been considered in order to get this property certified:

- It can be done by trying to access the configuration file with the right user, and with another user without the right privileges (this strategy will need for sure a hook in the WeLight Virtual Machine)
- Checking the owner of the settings file and its permissions, and compare them to the ones registered originally. This information has to be an input of the certification process, so it has to be stored somewhere, maybe it could appear on the certificate like sort of precondition.
- Some research will be done with regard to Linux and specific daemons able to register changes in file permissions. If that daemon registers changes in the settings file permissions there is a lack of security. Some Linux distributions support SELinux [3], a security module of the Linux kernel that provides with mechanisms that allow to implement security policies for access control. Some research will be carried out in order to find out if SELinux can detect changes on the settings file and who performed those changes, and eventually check whether that user was entitled to do that.
- Monitoring techniques could be also considered to certify this property. An event when a user attempts to access a file can be captured. Then it is checked whether or not the access was granted and if it had actually to be granted. Depending on the outcome of those checks, the ToC can be issued a certificate.

WeLight database driver

SQL injection is a kind of attack where a malicious user is able to execute arbitrary SQL code on a database. That could result in records being deleted or data leakage.

The goal is to certify that the WeLight database driver is able to resist this kind of attacks, what means that it manages to escape them.

Despite not being accomplished yet, this security property has been broadly discussed within the consortium.

At the beginning, it was considered that by using Django querysets [4][5] the resulting SQL would be properly escaped by the underlying database driver. Django [4] is a Python web framework that ease the bulding of a webapp. WeLight is developed by using Django. However, Django also gives developers power to write raw queries or execute custom SQL code.

One of the features of Django is its ORM. A good Django webapp does not include any SQL code because each query is managed by an specific database backend (Sqlite, Oracle, MySQL…). Such backend builds and sanitize the raw SQL query that finally will be performed in the real database engine.

The protection against SQL injection can be logged in two points. The first one is using Apache *mod_security* that log possible attacks, one of them is SQL injection, and, if enabled, can also block the malicious request. The second one is enabling the Django debug mode that allows to write the raw SQL query generated from each queryset. This mode generates a huge quantity of data.

The mechanism used to check this protection lies in searching all over the code of the server application so as to verify that no methods executing SQL code in a direct manner are being used in the queries.

The existence of those commands can be checked by executing the following:

```
$ ack-grep --python  "(\.extra\(.+\))|(\.raw\(.+\))|(\.execute\(.+\))"  /opt/welight/app
```

This was discarded since using logs implies that there has been some preprocessing and what would be ideal is to work with something 'pure' not having been processed at all. Because of that, the discussion started to point to the possibility of capturing packets, prior to any processing.

This has to do with another possibility that was also considered. This was about certifying the correct rejection of SQL injection attacks by means of monitoring techniques. To do this, an agent would be implemented in order to capture events of attempts to interact with the database. What would be actually captured are packets (IP datagrams). The relevant information would be extracted and parsed to an XML file. This file would be published on an event bus (Open Fire) so that the Monitor Manager (playing the role of a subscriber) would be able to retrieve it, analyze the evidences and eventually decide whether issue the corresponding certificate. This possibility was in the end ruled out since it was concluded that nothing relevant could be extracted from the datagrams aiming at certifying this property.

This method was not followed in the end, but it was found that the methodology was totally appropriate to certify the correct operation of the VPN (section 3.1.1) and the correct redirection of HTTP requests to the HTTPS port to ensure that the communications are secure (section 3.1.2)

Other methods rely on pattern identification. An agent able to identify if the SQL code that is being received follows a normal pattern or, on the contrary, something might be wrong on it could be implemented. By means of pattern identification SQL injections could be detected.

The next step would be to analyze in depth likely solutions to certify this property by means of testing techniques.

3.2.3. Storage Freshness

This property means that any attempt to retrieve a data object from storage in the ToC is guaranteed to return the most up-to-date version of the data object.

In consequence, the system must guarantee that the metrics retrieved should always correspond to the last committed version of the data.

First of all, it is necessary to explain the process by which the WeLight application receives the information about electrical consumption extracted by the LCU´s.

The application requires data to the LCU at intervals of 5 minutes. If the LCU is available (has connectivity), it will respond and there could be more than one sample in the response depending on certain conditions. Within the packet sent as a response, the timestamp can be found, somehow encrypted. When received, the timestamp is translated into plain text and stored in this manner on the database.

There can be two ways to certify this property: on the one hand a query can be sent to the database and it can be checked whether the timestamp is fresh enough (if the timestamp is older than the period established to capture the data, i.e. 5 minutes, it is not fresh) by comparing it with the system clock. On the other hand, the packets circulating between the web server and the LCU´s can be captured. This is a complicated option due to several reasons: because between the server and the LCU not only power consumption samples are communicated, but also other kind of information such as on/off controls; because the timestamp is encrypted and its translating process is complex and a sensitive information;

and another extra difficulty is the fact that there are several LCU´s sending information and it should be checked in each case which LCU data comes from.

Apart from all of this, it can be also monitored whether there was a particular period when packets stopped being sent or received.

3.2.4. Data Durability

This property matches **DSI:durability:percentage-durability**. This security property describes the durability for stored data: the probability (expressed as a percentage) that a data object will remain unaltered over a certain period

In consequence, this is about the system guaranteeing the durability for stored data and getting that property certified.

In order to ensure the compliance of this property, a log containing the deleted database rows will be added. As mentioned within this deliverable, these rows contain data regarding electrical consumption measures. By checking this log it can be ascertained whether this property is being fulfilled. It will be necessary to define the time window for which it is required to keep that information. That will define an estimation of the required disk space for the log.

Works on this property will be carried out in year 3. It is not decided yet which technique will be used for the certification process. One possibility is to monitor the database engine and take advantage of the logs provided by it to retrieve information about deleted rows. This does not seem to be compliant with techniques based on monitoring to extract evidences, since in the case of monitoring the data is preferred ‘pure’ without any kind of previous process. Using logs unavoidably implies that processing taking place. So this property will be likely to be certified by using **testing** techniques.

Progresses in this regard will be informed in D6.5.

3.2.5. Software Integrity Protection

The purpose is to protect the ToC against the alteration of its software. The property could be considered as ‘verified’ if the software is executed in tamper-proof hardware security module.

A related property is **AIS:integrity:software-integrity-detection.** Any modification of the software running on the ToC is detected and reported. The property could be considered as ‘verified’ if a remote attestation mechanism allows verifying the integrity of the software on the ToC.

In consequence, the system must ensure that the software running on the Light Control Unit has not been modified (false data injection). This check must be done from the server side in order to avoid forgery in the signatures of the files composing the code.

The method followed to certify this property is similar to the case shown in section 3.2.2. A command will be executed by means of an SSH connection to check the signature of the files and compare it to the one stored in the web server database.

The command that will be used may be something similar to what follows:

```
$ find /path/to/code/ -type f -name "*.py" -exec md5sum {} + | awk '{print $1}' | sort |
md5sum
 f1fe1b965c8a3b17232371a35fa09f44  -
```

As with *File System Encryption Security*, the most likely way to certify this security property will be by means of monitoring techniques, implementing an agent able to capture the moment when a command similar to the one shown above is executed and extract the relevant information so as to send it as evidences of the security property being or not fulfilled.

3.2.6. Measures Integrity

This property matches **AIS: integrity:data-alteration-prevention**.

This means that the data stored by the ToC is protected against unauthorized alteration.

This can be also related to **AIS:integrity:data-alteration-detection**.

This means that any alteration of data stored by the ToC is detected.

The property could be considered as 'verified' if data from the ToC is compared with a reference fingerprint (hash), which is itself stored in a secure location. Any alteration of the ToC can therefore be detected.

In consequence, the system must guarantee that data provided by the service is not altered either in transit or in the database by an unauthorized user.

Checking this property is a bit complex. It involves the implementation of a process by which the information on the LCU is signed and the subsequent checking at server side.

The process would involve the steps explained above:

1) Assign a unique key to each LCU.
2) Use that unique key to sign the data that the LCU sends to the server.
3) Add the signed data to the message containing the data themselves and sign it.
4) Once the message is received, extract the information inside and separate the signed and plane version of those data.
5) Sign the received plain data with the unique LCU key.
6) Check the coincidence between the data that was signed at server side and the data signed at client side. If those data are the same, the signed version should be also the same.

3.2.7. Availability of information (Quality of Service)

This property matches **BCR:availability:percentage-of-uptime**

This is a periodical verification of the service being active and operational. There will be a call to an API in order to get LCU percentage of activity during the last 24 hours.

Actually, the response (or not) to the request shows the availability of the webserver. Besides, the content of such response shows the availability of the LCU´s.

3.2.8. Non-repudiation of origin

This is about verifying that when a user sends a request to a cloud provider, none of the involved parties can deny having participated in the transaction. By certifying this property support to resolution of disputes about the fact of given data being or not generated in the system can be provided.

4. Conclusions and next steps

Below the main achievements are enumerated:

- The Smart Cities Pilot has been adapted to be used within CUMULUS testbed. A couple of instances, one running over VMWare and another one running over OpenStack have been deployed in the testbed located at the University of Malaga. The one over VMWare has been used to carry out the work regarding certification with monitoring-based techniques, and the one over OpenStack has been used to obtain certificates based on testing techniques. Along with this deliverable, a confidential annex including the user guide and the deployment guide is included.
- WeLight has been notably improved as far as security is concerned by developing a set of security mechanisms in the framework of the CUMULUS project. In such sense, the product is more competitive and appealing to be marketed. These security mechanisms cover the main security requirements expressed in D6.1. Besides, the security properties expressed in the same deliverable are also covered except for those having to do with non-repudiation of the origin of client reports and client alerts. These security properties will be covered as a goal to be reached before the end of the project.
- Six security properties have been proposed as proofs of concept to test the certification mechanisms. Testing and monitoring-based techniques, with the support of TC in some cases have been used. In addition, tests performed at different layers of the cloud stack have been combined in order to issue multilayer certificates.
- There is an ongoing work investigating other ToCs and security properties that could be certified. We note that some of the corresponding security mechanisms are already in place.

The next steps till the end of the project are the following:

- Several security properties will be considered to be issued by hybrid and incremental certification models. The progresses made will be reported in D6.5 and D6.6.
- In order to get these new properties certified, it will be necessary to develop new security mechanisms. This will benefit WeLight, which will become even more secure.

References

[1] Keyzcar cryptographic toolkit: http://www.keyzcar.org

[2] Raspberry PI: http://www.raspberrypi.org

[3] SELinux: http://selinuxproject.org

[4] Django: https://www.djangoproject.com

[5] Django querysets: https://docs.djangoproject.com/en/dev/ref/models/querysets/

[6] CUMULUS D3.2 "Core Certification Mechanisms v2" section 3.3.2

[7] CUMULUS D3.2 "Core Certification Mechanisms v2" section 3.3.3

[8] CUMULUS D3.2 "Core Certification Mechanisms v2" section 3.3

[9] CUMULUS D3.2 "Core Certification Mechanisms v2" section 4.2

[10] SUSE Linux Enterprise Server 11 SP3: https://www.suse.com/documentation/sles11/

[11] IBM Knowledge Center: *Securing sensitive files with TPM keys*: http://www-01.ibm.com/support/knowledgecenter/linuxonibm/liaai/tpm/liaaitpmstart.htm

[12] Red Hat, Inc. :Red Hat Enterprise Linux 5.3, Release Notes: http://www.centos.org/docs/5/html/5.3/pdf/Release_Notes.pdf

[13] CUMULUS D6.1 "Specification of pilots scenarios and requirements" section 3.5.2

[14] CUMULUS D6.1 "Specification of pilots scenarios and requirements" section 5.3.3

[15] CUMULUS D3.2 "Core Certification Mechanisms v2" section 4.2.1

[16] CUMULUS D3.2 "Core Certification Mechanisms v2" section 4.2.3