Horizon 2020 Program (2014-2020)

Cybersecurity, Trustworthy ICT Research & Innovation Actions
Security-by-design for end-to-end security
ICT 32-2014

Secure Hardware-Software Architectures for
Robust Computing Systems [†]

Deliverable D2.1: SHARCS Applications and framework requirements for secure-by-design systems

Abstract: This document describes the various applications used as case studies within SHARCS, as well as their security risks, threats and security-related requirements.

Contractual Date of Delivery	Month 12
Actual Date of Delivery	Month 12
Deliverable Dissemination Level	Public
Editor	Christos Strydis
Contributors	All SHARCS partners
Quality Assurance	Ioannis Sourdis, Vassilis Prevelakis

[†] The research leading to these results has received funding from the European Union Horizon 2020 Program (2014-2020) under grant agreement n° 644571.

The SHARCS Consortium

Foundation for Research and Technology – Hellas	Coordinator	Greece
Vrije Universiteit Amsterdam	Principal Contractor	The Netherlands
Chalmers Tekniska Högskola	Principal Contractor	Sweden
Technische Universität Braunschweig	Principal Contractor	Germany
Neurasmus BV	Principal Contractor	The Netherlands
OnApp Limited	Principal Contractor	United Kingdom
IBM - Science and Technology LTD	Principal Contractor	Israel
Elektrobit Automotive GMBH	Principal Contractor	Germany

Document Revisions & Quality Assurance

Internal Reviewers

1. Ioannis Sourdis (CHAL)
2. Vassilis Prevelakis (TUBS)

Revisions

Ver.	Date	By	Overview
1.0.3	20/12/2015	Christos Strydis (Neurasmus)	Final edits and re-release to coordinator.
1.0.2	18/12/2015	Thomas Kamm (Electrobit)	Minor enhancements to Chapter 3.
1.0.1	20/11/2015	All app partners	Final corrections and release to coordinator.
1.0.0	12/11/2015	All app partners	Further edits based on QA review.
0.1.4	6/10/2015	Christos Strydis (Neurasmus)	Final edits to deliverable and release to QA.
0.1.3	5/10/2015	Christos Strydis (Neurasmus)	Modifications to Implant use case.
0.1.2	2/10/2015	Thomas Kamm (Elektrobit)	Modifications to Automotive use case.
0.1.1	1/10/2015	John Thomson (OnApp)	Modifications to Cloud use case.
0.1.0	17/9/2015	Christos Strydis (Neurasmus)	Intro text fixed. First draft ready.
0.0.8	17/9/2015	Thomas Kamm (Elektrobit)	Security-related text for Cloud use case updated.
0.0.7	16/9/2015	John Thomson (OnApp)	Figures added, text improved in Cloud use case.
0.0.6	14/9/2015	Thomas Kamm (Elektrobit)	Text improved in Automotive use case.
0.0.5	10/9/2015	Christos Strydis (Neurasmus)	Implant use case and general text updated.
0.0.4	11/8/2015	Christos Strydis (Neurasmus)	Document structure updated.
0.0.3	30/7/2015	Martin Böhner (Elektrobit)	Automotive use case added and figures.
0.0.2	13/7/2015	John Thomson (OnApp)	Cloud use case converted to LaTeX.
0.0.1	7/7/2015	Christos Strydis (Neurasmus)	Outline of the document created.

Contents

List of Figures

Acronyms

AAA	Authentication, authorisation and accounting
AD	Application Developer
AO	appliance owner
API	application programming interface
ASIP	Application-Specific Instruction Processor
CERIAS	Center for Education and Research in Information Assurance and Security
CIANA	Confidentiality, Integrity, Availability, Non-Repudiation and Authentication
CP	Cloud Provider
CP	Control Panel
CPP	Cloud Platform Provider
CPU	Central Processing Unit
CSA	Cloud Security Alliance
CSIG	cloud select industry group
CSIRT	Computer systems incident report team
CSP	Cloud Service Provider
CSRF	Cross-site request forgery

CVE Common Vulnerabilities and Exposure

CVSS Common Vulnerability Scoring System

DCO datacenter owner

DoS Denial of Service

DREAD Damage, Reproducibility, Exploitability, Affected users + Discoverability

EEG Electro-Encephalo-Gram

ENISA European Network and Information Security Agency

EU end user

FIRST Forum of Incident Response and Security Teams

HIPAA Health Insurance Portability and Accountability Act

HV hypervisor

I/O Input / Output

IaaS infrastructure as a service

IMD Implantable Medical Device

IOP Input/Output oPerations per second

IP Internet Protocol

KVM Kernel-based Virtual Machine

MAC Message-Authentication Code

NVD National Vulnerability Database

OS Operating System

OWASP Open Web Application Security Project

QoS Quality of Service

RAM Random-Access Memory

RPM RPM Package Manager

SaaS Software as a Service

SAN Storage Area Network

SCAP Security Content Automation Protocol

SLA Service Level Agreement

SSAE Statement on Standards for Attestation Engagements

SSH Secure Shell

SSL Secure Socket Layer

STRIDE Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service, Elevation of Privilege

TLS Transport-Layer Security

TPM Trusted platform module

US-CERT United States Computer Emergency Readiness Team

VM virtual machine

VNC Virtual Network Computing

XML Extensible Markup Language

XSA Xen Security Advisory

1 Introduction

1.1 Scope

This document describes deliverable D2.1, the outcome of task T2.1 which is involved with the specification of the security requirements of the three SHARCS use-cases: the Implant application (Chapter 2), the Automotive application (Chapter 3) and the Cloud application (Chapter 4).

To facilitate this deliverable, the security partners have initially laid out an explanation of the general security features necessary for secure-by-design applications and have given the guidelines on how the application partners (Neurasmus, Elektrobit, OnApp) should characterise their respective applications. Each application partner, then, analyzed their application and defined the features relevant and available to the SHARCS framework, so as to achieve – at the end of the project – end-to-end security for those applications. The resulting set of features has led to a detailed list of security requirements that are the basis for the work in Work-packages 3 (WP3) and 4 (WP4).

1.2 Document structure

In order to compile the list of security requirements, in this report each SHARCS application provider has detailed their application as a use-case to be enhanced in SHARCS with end-to-end security. In more detail, each application partner has abided by the following structure in filling in this document:

- **Application description:** In this section, the system overview of each application is given, along with its software and hardware components and their relationship.

- **Security evaluation:** Security-related information is given such as the security threats, the threat model and the attack scenarios of each application.
- **Requirements:** The security-related requirements of the application are presented, as set out by the respective SHARCS application providers. The ultimate objective of these requirements is to deliver and demonstrate end-to-end security.

On delivering these security requirements, the SHARCS partners will be able to proceed with the definition, design and implementation of the hardware (WP3) and software (WP4) security provisions to comprise the SHARCS framework. This document will also further assist in defining the metrics and benchmarks (D2.2) for evaluating the SHARCS security techniques during application evaluation in WP5.

1.3 End-to-end security objective

Generally speaking, end-to-end security is a complex problem which, further, lacks a complete definition. Lacking a formal method for tackling end-to-end security, the SHARCS consortium has decided to *accept as satisfactory end-to-end security the fulfilment of a set of security requirements* per application use-case, as outlined in the section above.

Figure 1.1 shows the general approach the SHARCS partners will follow for achieving end-to-end security. Deliverable D2.1 (this document) will specify and detail the security requirements ('SR' in the figure) per use-case, the fulfilment of which will deliver end-to-end security. SHARCS deliverable D2.2 will – among others – enumerate the potential threats ('TH' in the figure) which can lead to violation of the security requirements. Finally, deliverables D3.1 and D4.1 will compile an extensive list of all possible defense techniques ('DT' in the figure) in hardware and/or in software, respectively, that can be employed within SHARCS for tackling the above attack scenarios. In principle, each security requirement can be affected by any number of attacks each of which can – in turn – be tackled by any number of security techniques.

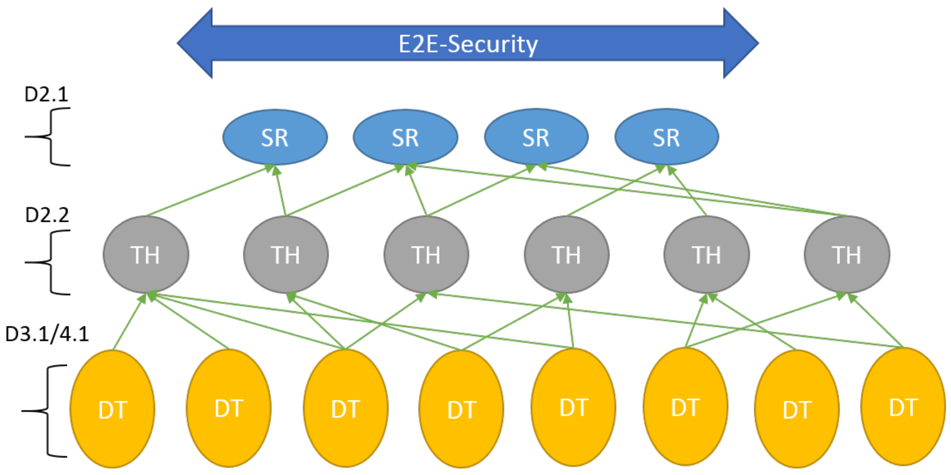

Figure 1.1: The SHARCS approach to tackling end-to-end security in the project will be achieved by respecting a set of security requirements (SR) per application use-case. Effort, then, will be put for devising and implementing defense techniques (DT) for counteracting various threats (TH) that can lead to violation of the set security requirements.

2 Implant application

Epilepsy is a group of short- or long-term neurological conditions which are characterised by epileptic seizures. Various treatments have been proposed, mostly revolving around periodic stimulation of the Vagus Nerve, the thalamic nuclei or deep-brain stimulation. It has been shown that these treatments may reduce the frequency of seizures in some patients but may result in several complications, including permanent damage. Only recently, treatments where stimulation is applied responsively (rather than periodically) using a seizure-detection method have been proposed.

The Implant Application is a novel, closed-loop, fully implantable neuromodulator that senses Electro-Encephalo-Gram (EEG) and single-neuron recordings, detects seizures before they manifest, and prevents seizure manifestation through highly selective optogenetic (or electric) stimulation of cerebellar neurons. Although implant functionality is autonomous, the implant should also communicate with the outside world for overall implant control (e.g. recalibration) as well as for sending patient-monitoring information to the patient, the doctor and so on. Figure 2.1 illustrates the overall implant application.

A prototype of the implantable neuromodulator has already been successfully tested for the treatment of absence epilepsy in mice by Neurasmus BV [6]. In Figure 2.2, the prototype built with off-the-shelf components is illustrated. The implant is smart, adaptive and autonomous but it can also work under the remote control of an external handheld reader device, e.g. a smartphone. The handheld can perform various operations such as implant control, (re)calibration and data logging. In turn, the handheld, acting also as a wireless gateway, has to be able to communicate with cloud services such as a medical data repository, a compute medical cluster for off-line heavy-load computations, and so on.

The implant use case represents a research field which is quite dymanic and is being actively explored at the moment. As a result, there are vast

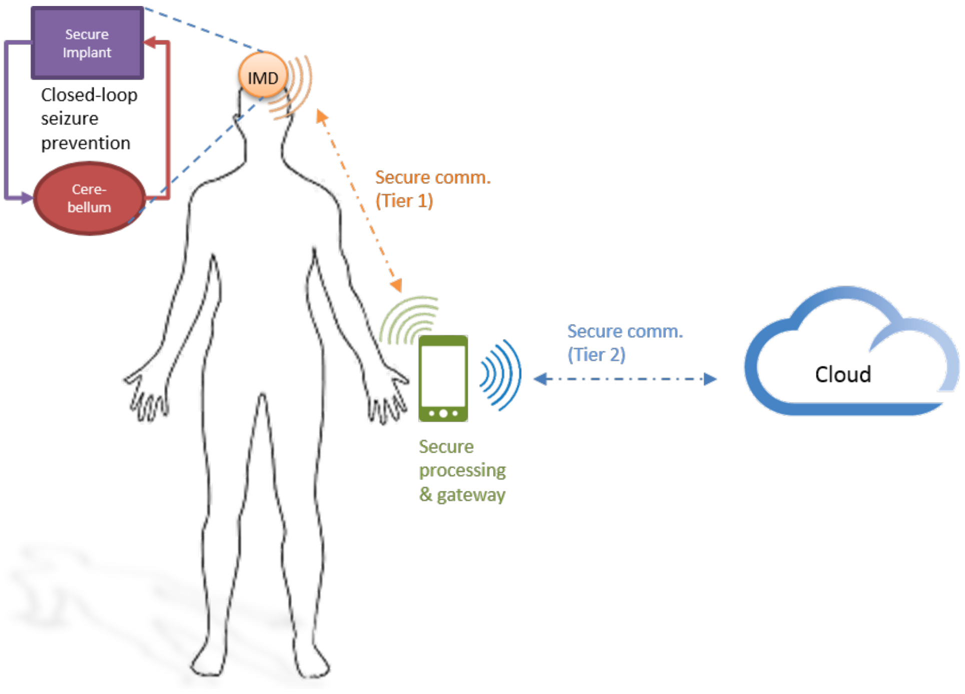

Figure 2.1: Overview of the implant application.

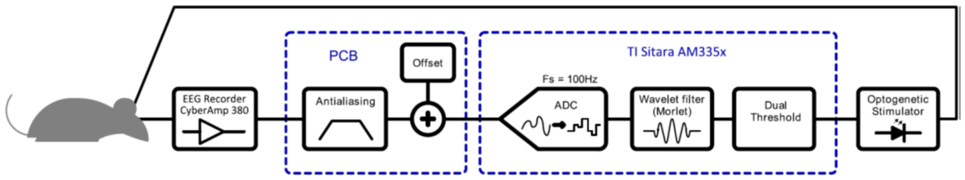

Figure 2.2: Block diagram of closed-loop, seizure-prevention prototype

design alternatives to explore both in terms of hardware and of software. In effect, this use case does not come with the constraints of a perfectly established domain and lends itself naturally for implementing the SHARCS 'clean-slate' model highlighted in Figure 2.3. The potential interventions to the implant use case are many, yet so are the opportunities for achieving high security levels.

2.1 System description

The seizure-prevention neuromodulator is a closed-loop system which achieves its functionality by stimulating the brain if it exhibits seizure-related electrical activity. The neuromodulator is implemented in the Neurasmus SoC, depicted in Figure 2.4, which consists of a number of digital and analog

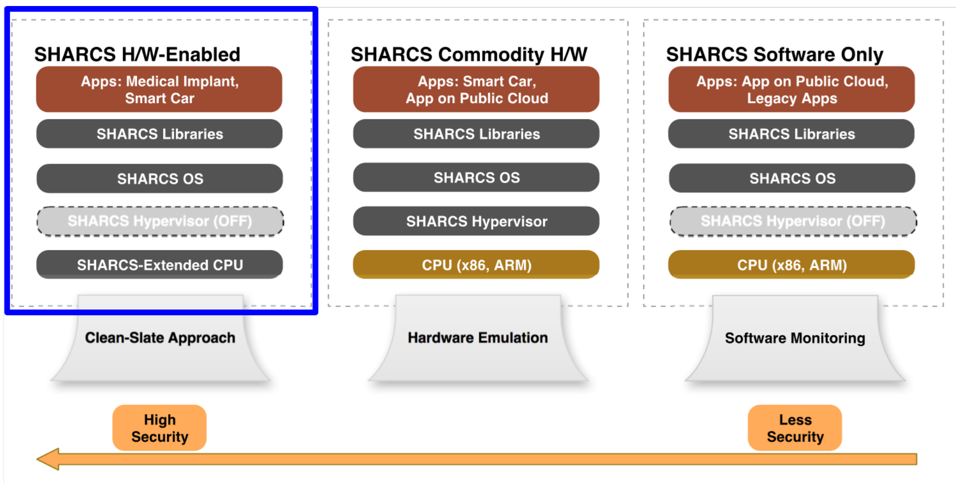

Figure 2.3: SHARCS-framework operational model, highlighting the areas relevant to the implant application

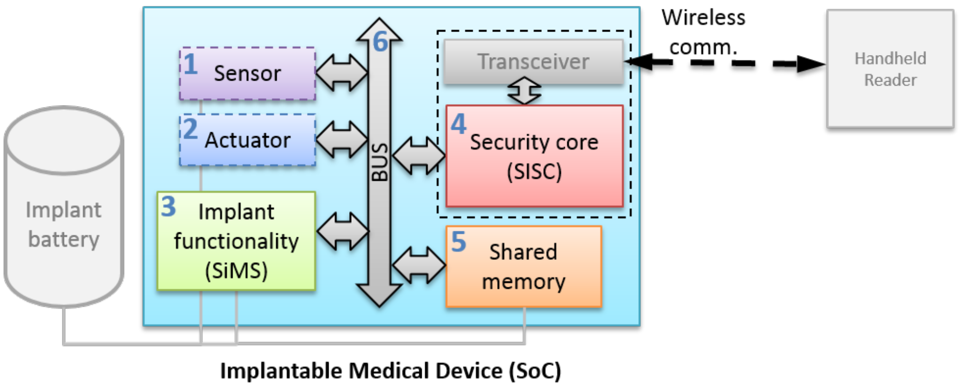

Figure 2.4: Neurasmus SoC implementing the implant application

components. Within SHARCS, we set the application boundary to contain all digital components plus wireless communication to/from the SoC. That is, we will consider the following components:

1. **Sensor (digital):** This module obtains a digitised version of the sensory recording of an implanted electrode (saved in a register or memory location). It is a hardware block responsible for providing the SiMS-processor with correct and fresh data-samples at real-time every 10 ms. Within SHARCS, the digitised input to the sensor module will be emulated in hardware.

2. **Actuator (digital):** This hardware module forms the main output of the SoC. It is responsible for starting and stopping stimulation on the

implanted electrode, based on a command received from the SiMS-processor. Within SHARCS, we include the memory location that the actuator reads as part of the application boundary.

3. **SiMS processor (main implant functionality):** This module consists of three hardware components (the SiMS-processor with private instruction and data memory) and a software (the implant functionality) running on it. This module processes the input from the sensor and determines if the input shows seizure-related electrical activity. If so, it sends a message to the actuator module to start / stop stimulating.

4. **SISC processor (secure communication):** The SISC processor is tasked with handling all (secured) communication to and from the SoC, without disrupting the main Implantable Medical Device (IMD) functionality performed by the SiMS core. It has its own (private) instruction- and data-memory blocks.

5. **Shared-memory block:** This is a hardware component used for logging data originating from the SiMS (start and stop-time of seizure-activity) and SISC processor (data exchanged through the wireless communication).

6. **SoC interconnect (bus):** This hardware component is responsible for allowing communication between the components in the SoC.

In other words, the application boundary is the digital SoC including the wireless communication link. It does not include the implant transceiver and the handheld device (reader) – denoted with greyed-out boxes in Figure 2.4 – or any analog components. By the same assumption, tampering with the input (thus, affecting the output) data is a concern only at the level of the input/output registers (i.e. digitised data).

The SoC performs two main operations: Autonomous seizure prevention (main functionality) and external-device control. We describe these two operations next, detailing the SoC-components involved and the communication between them.

The prototype, at this stage, implements only the basic implant functionality. That is, it does not include a complete communication protocol for supporting treatment adjustment, reading of patient data, etc. In short, it doesn't include all necessary provisions that would make it vulnerable to security attacks. Thus, part of the work for this use-case will be the completion of the system functionality, according to the specifications presented in the next sections.

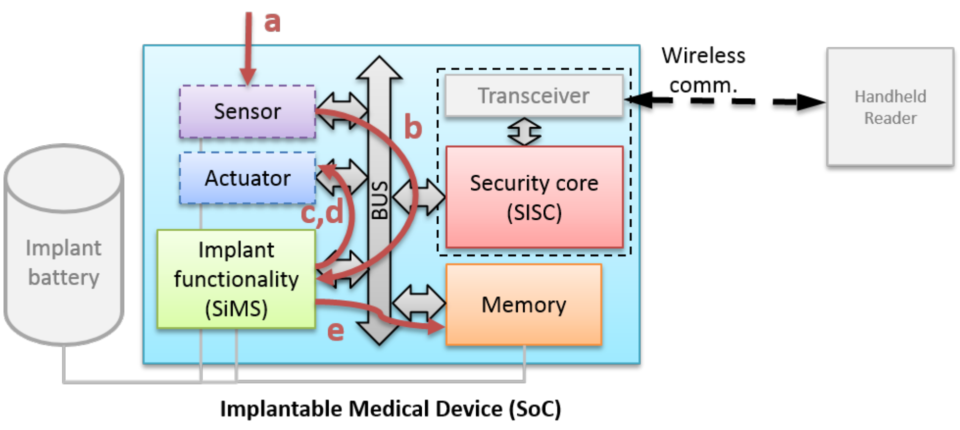

Figure 2.5: Use-case 1: Autonomous seizure prevention

2.1.1 Autonomous seizure prevention

The main implant functionality is the autonomous prevention of seizures, which consists of two steps: 1) The real-time monitoring of the activity in the brain; and 2) The actuation in case a seizure is detected. Real-time monitoring is accomplished through iteratively sampling brain activity (using the sensor) and a seizure-detection function executed on the SiMS processor (see Figure 2.5):

a) Every 10 ms, the sensor obtains a new sample and sends it to the SiMS processor;

b) For every sample, the SiMS processor evaluates if a seizure is manifesting. If not, SiMS goes back to idle state; (in a future version, SiMS will enter a low-power state);

Once a seizure is detected, the following steps are executed:

c) The SiMS processor sends a “start-stimulation” command to the actuator (alternatively, a “stop-stimulation” command is transmitted if a seizure has successfully been suppressed);

d) The actuator stimulates until a “stop-stimulation” command is issued by the SiMS processor. As detection is based on a thresholding filter, stimulation duration depends on the (sensor-provided) EEG input signal. In the current implant prototype, there is no risk of race conditions since functionality is deterministic. However, in the final SHARCS system, the SiMS processor has to be prioritised over other modules talking over the same bus (e.g. through the bus arbiter) so as to ensure that the “stop-stimulation” signal is received and processed in time;

e) The SiMS processor registers the seizure event in the data log (shared memory).

Real-time performance is, at this moment, guaranteed through using a rather simple closed-loop-control system (see Figure 2.2) which does not rely on interrupts for its operation and executes a rather deterministic seizure-detection algorithm. The full-fledged version of the system – including a security protocol, both SiMS and SISC operating in unison and more sophisticated external-reader interaction – will, however, require more active guarantees for system real-time performance like, for instance, watchdog times. Failing this requirement can lead to severe security and safety issues such as delivering improper or untimely stimulation, failing to protect patient-data privacy, and so on. For those reasons, the real-time requirement is effectively captured by one of the application requirements listed in Section 2.3.1.

2.1.2 External-device control

It is possible for the external (handheld) reader to monitor the neuromodulator or patient (checking data-logs), or trigger a configuration (update treatment settings or code-maintenance) by communicating over a secure (wireless) channel. The security settings currently implemented for this channel are briefly presented in Section 2.1.4. For either one of the reader functions, the external reader must first establish two-way authentication with the implant:

a) The external reader sends a communication request to the transceiver;

b) The transceiver forwards the message to the SISC core, which authenticates the reader (if provided with the correct credentials).

Once authenticated, the external reader may (depending on his or her privilege levels) request data logs or configure the main implant functionality. For data read-out, the following steps are executed (see Figure 2.6):

c) The external reader requests a data-readout from the implant by sending a message over a secure channel;

d) The transceiver forwards the message to the SISC core, which recovers the message;

e) The SISC processor obtains the requested data from the shared memory;

f) The data is transmitted to the external reader (through the transceiver) over the secure channel.

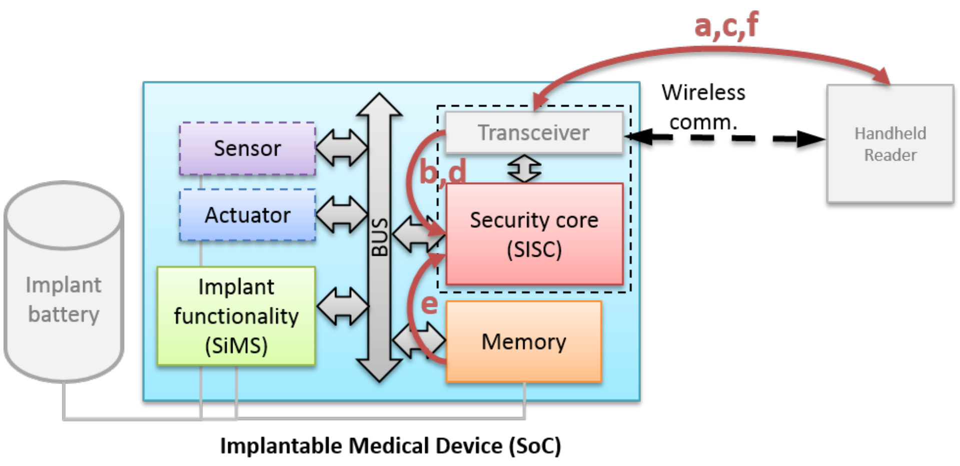

Figure 2.6: Use-case 2a: External device control (data read-out)

For configuration, the following steps are performed (see Figure 2.7):

c) The external reader configures the implant by sending a configuration message over a secure (wireless) channel;

d) The transceiver forwards the message to the SISC core, which decrypts the message;

e) The SISC processor triggers the requested (re-)configuration on the SoC;

f) The SISC processor replies to the external reader once the (re)configuration is complete.

2.1.3 User roles

For the implant application, three user roles can be considered in the context of SHARCS: “Patient”, “Physician” and “Technician”, the latter roles having gradually higher permissions than the former ones. The list of permissible actions inside the implant are generally as follows:

(I) Read out data related to the seizure-prevention activity to an external monitoring device. For this implant, such data would be log entries for manifested seizure events and corresponding blocks from the implant.

(II) Read or modify configuration parameters affecting the seizure-prevention functionality (e.g. detection sensitivity). This in effect means access to a few protected SiMS-data-memory or shared-memory or configuration-register entries in the implant.

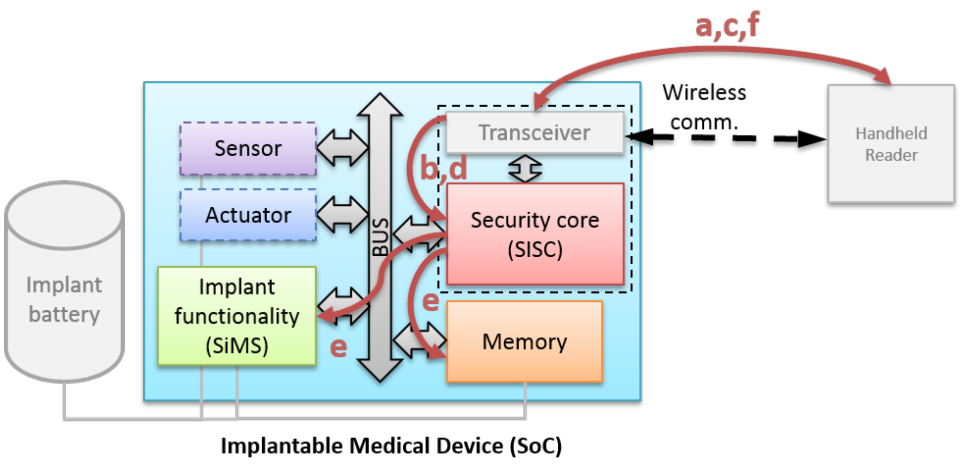

Figure 2.7: Use-case 2b: External device control (implant configuration)

(III) Turn on and off the implant.

(IV) Flash the SiMS/SISC program memory with different binaries. Such flashing can either be used for upgrading the implant functionality or security, for eliminating manufacture-time software bugs, for adjusting implant functionality to patient particularities (over time) or – even – for creating debugging and diagnostics scenarios.

(V) Read or write the shared-memory contents, the implant control registers etc. Since all implant peripherals are memory-mapped, control of the shared memory in effect permits advanced modes of diagnostics, testing and debugging.

Based on the above considered implant-side actions, the 3 user roles shall have the following permissions:

UR-1.1 **Patient:** The "patient" role involves the carrier of the implant himself/herself as well as potentially close family members (e.g. spouse, children) as care-givers to the patient and early responders in case of emergency. This role permits only action (I).

UR-1.2 **Physician:** The "physician" role is reserved for the doctor (i.e. treating physician) personally monitoring the patient's progress. This role is allowed to do what the "patient" role can – i.e. read data off of the device e.g. in his work computer – and is also allowed to effect some changes in the implant's seizure-prevention functionality or, also, shut down the device, if the need arises. Thus, this role permits actions (I), (II) and (III).

Table 2.1: Implant-application user roles

	Roles	**Permission level**	**Permissions**
UR-1.1	Patient	Lowest	Read application-related data ((I))
UR-1.2	Physician		Read/modify application-related data; switch device on/off ((I), (II), (III))
UR-1.3	Technician	Higher	Read/modify all implant data; switch device on/off; update device firmware ((I), (II), (III), (IV), (V))

UR-1.3 Technician: The "technician" role refers to the permissions of the medical technician or bioengineer responsible for keeping the implant in check, performing periodic maintenance tasks, and so on. Thus, this role permits all actions (I), (II), (III), (IV) and (V).

Access control is carried out simply (through control checks) by the standalone application, so no extra access-control module is used. In summary, the following user roles can be considered for the implant application (see Table 2.1).

2.1.4 Scope - Existing security

For the Neurasmus implant use case, a system architecture has already been designed [5] where security and main-implant functionality are made completely decoupled by running the tasks onto two separate Application-Specific Instruction Processors (ASIPs). Main implant functionality is handled in the SiMS core. Wireless communication goes through the SISC core, which runs an energy-efficient security protocol. What is more, the security core is powered by RF-harvested energy until it performs external-reader authentication, providing an elegant defense mechanism against *battery Denial of Service (DoS)* and other, more common attacks. The system architecture (already presented in Figure 2.4) achieves defense against unauthorised accesses having *zero energy cost* when running entity authentication through harvested RF-energy from the requesting entity. In all other successfully authenticated accesses, the implant architecture achieves secure data exchange without affecting the performance of the main implant functionality.

For achieving secure communication between the implant and an external reader we have implemented a lightweight security protocol based on the ISO/IEC 9798 (Part 2) standard [3]. Our proposed scheme is based on the fourth protocol of this standard and currently implements a four-pass, mutual-authentication protocol. It employs a *symmetric cipher* for message encryption. For key management, an *offline* key-distribution mechanism has been assumed. Entity authentication and message integrity are covered by using a Message-Authentication Code (MAC) for the data transfer. *Freshness* and protection against *replay attacks* is guaranteed through use of *random numbers*, which are generated on-the-fly during protocol execution. Further implementation details can be found in [5].

2.2 Security evaluation

Based on the neuromodulator functionality and components described in the previous section, we may now evaluate the security of the Neurasmus SoC. First, we describe the threats associated to a security breach, as well as the threat model which specifies the assumptions under which we strive to guarantee security.

2.2.1 Security threats

Security threats describe particular sources or means through which particular types of attack can be mounted. Here, we discuss a detailed list of threats in decreasing order of importance, including the *key-security concepts* related to this threat and the *components and attack scenarios* involved which result in the manifestation of this threat. For specifying the security concepts associated with each threat, we have chosen to comply – across all three SHARCS use-cases – with the Confidentiality, Integrity, Availability, Non-Repudiation and Authentication (CIANA) formalism.

2.2.1.1 TH-1.1: Modification of IMD operation

The main functionality of the implant application is to provide closed-loop, selective real-time stimulation of neurons in order to prevent seizure manifestation. Modification of operation may result in the prevention of stimulation during treatment (that is, no treatment is provided), as well as over-stimulation of the tissue (potentially resulting in tissue damage) or also in device operational-lifetime shortening or device breakdown (e.g. broken vias). These threats can come about directly – e.g. by having the SiMS or SISC cores execute additional code, fiddling with mission-critical application timers etc. – or indirectly – e.g. through increasing the device's power and energy consumption by executing redundant code.

- **Key security concepts / CIANA:** Integrity, Availability (as the the device no longer works as intended), Authentication.
- **Scenarios and involved components:**

1. The SiMS processor is halted or its functionality (i.e. when to stimulate) is altered.
2. The sensor reads incorrect values, resulting in incorrect calculations by the SiMS processor.
3. The actuator is fed incorrect values, resulting in wrong stimulation activations.
4. The SISC processor triggers a binary-code change in the SiMS processor, resulting in incorrect functionality.
5. The timing, frequency or duration of treatment could be modified (e.g. clock-timing or frequency-scaling attack). If the clock is changed, the SiMS is still working as expected but the system is tweaked to do too much or too little or at the wrong time in stimulating the patient.
6. The SISC or SiMS processors are forced to execute additional code, loop indefinitely or operate at a higher frequency, resulting in device hotspots and battery waste.

2.2.1.2 TH-1.2: Data-log manipulation

Unauthorised data manipulation (data forging) can indirectly lead to patient/doctor misinformation and subsequent wrong-treatment delivery.

- **Key security concepts / CIANA:** Confidentiality, Integrity (manipulation), Non-repudiation, Authentication.
- **Scenarios and involved components:**
  1. The entries in the shared-memory module are modified;
  2. The communication between the SISC processor and external reader is unsecured, resulting in communication-packet manipulation.

2.2.1.3 TH-1.3: Data theft

Implant-located patient data is private and must be securely stored and transmitted. Data theft can indirectly lead to problems such as social segregation, extortion, blackmail and more.

- **Key security concepts / CIANA:** Confidentiality, Non-repudiation.
- **Scenarios and involved components:**
  1. The entries in the shared-memory module are stolen;
  2. The communication between the SISC processor and external reader is unsecured, resulting in data-leakage (overhearing).

2.2.2 Threat model

We report, next, the assumptions made under which security should be provisioned for the implant use case:

AS-1.1 There is only remote (nonphysical) access to the device (since it is implanted);

AS-1.2 The device is *fully shielded*, preventing electromagnetic interference;

AS-1.3 The authentication credentials are unknown to an adversary[1];

AS-1.4 The cryptographic cipher and security protocol are secure;

AS-1.5 An attacker can send arbitrary messages over the wireless link.

2.3 Requirements

Based on the threats described in the previous section, we may now define the (security-related) requirements of the device and application.

Achieving security for an implant can be divided into two main parts. The first part, present in every secure system, is to make the system resilient to attacks. The second part – not so common in other secure systems – is to make the implant very power- and energy-efficient. Excess power or energy consumption as well as unauthorised access could lead to compromising the implant-host's health and, even, to death. In this section we, thus, first describe application requirements set by Neurasmus and constraints which affect implant security indirectly. We, subsequently, expand this list with the essential, direct security features required by the device.

2.3.1 Application requirements

Being a mission-critical system, it is imperative that the implant functions within certain timing, energy and power margins during its complete operational lifetime. Except for affecting the fault-tolerance levels of the device, violating such margins can also indirectly lead to security compromises. Below, a detailed list of such margins is given. Please note that some of them have been characterised – at project onset – as hard and others as soft constraints.

AR-1.1 **Fixed sampling time (10 ms):** The implant application is configured for a sampling frequency of 100 Hz (10 ms per sample). Any deviation from this timing affects treatment adversely. (*Hard constraint*).

[1]Our existing as well as any new implant security protocols developed in SHARCS will aim at making acquisition or guessing of such credentials more difficult.

AR-1.2 Upper-bounded SiMS execution-time (10 ms): Given the fixed sampling time of 10 ms, each sample has to be processed (by the SiMS processor) in no more than 10 ms. (*Hard constraint*).

AR-1.3 Upper-bounded (instant., average) power consumption (10%): Increasing the power consumption results in increased heat dissipation (tissue damage) and reduced device lifetime. Given the importance of device lifetime (as invasive surgery is required to replace the device or battery), the power consumption should be increased by no more than 10%. (*Soft constraint*).

AR-1.4 Upper-bounded energy expenditure (10%): Similar to power consumption, increasing the energy expenditure will result in a reduction in device lifetime. (*Soft constraint*).

AR-1.5 Upper-bounded chip area (30%): Increasing the chip-area results in more expensive IC's. (*Soft constraint*).

The maximally allowed power, energy and area overheads given above are based on hands-on expertise with existing implant applications. A general rule of thumb is that power and energy are not allowed to increase by too much as we are dealing with ultra-low-power systems. Area, on the other hand, is somewhat more relaxed a constraint based on the fact that the largest contributor to implant size in fact is the battery pack, which takes up roughly 75% of the device [4]. Thus, a 30% or higher increase in chip area is not expected to cause any measurable increase in implant-package size, however, it may lead to more prolonged (and, thus, expensive) validation cycles. Therefore, we chose to limit it as well.

2.3.2 Security requirements

In view of the aforementioned threats, the threat model and the application-related requirements, essential security requirements of the system can now be set.

SR-1.1 Security compliance with extra-functional constraints: The hard constraints listed in Section 2.3.1 must be respected, regardless of any new security enhancements of SHARCS on the implant device. The soft constraints can be violated to the point that they do not compromise the hard constraints in the final SHARCS-enabled implant design.

SR-1.2 Security compliance with proper treatment delivery: The implant functionality is as follows: (a) sensory data are stored in dedicated entries in the shared-memory block; (b) detection software residing in the SiMS core scans the data for signs of a seizure onset; and (c) a proper command is sent to the stimulator output (actuator) through writing to other dedicated entries in the shared memory, whereupon stimulation is delivered to the brain tissue and seizure is suppressed. Safety timeouts (delivered e.g. through watchdog times) should be implemented to guarantee that stimulation to the tissue will be stopped after a maximum interval ($\leq 3\ sec$). The implant functionality is highly mission-critical and, as such, shall remain immutable at all times and under any modifications of the implants, e.g. for encompassing SHARCS security provisions.

SR-1.3 **Patient-data security and privacy:** Physiological data generated during normal operation of the implant are to be considered privately owned by the patient (user role: "Patient") and – after written consent – can be shared with his/her treating physician (user role: "Physician"). Moreover, patient private data shall be securely stored inside the implant data memories (private and shared) and securely transmitted on request across the wireless link to authorised monitoring devices.

SR-1.4 **Patient safety & device accessibility:** Patient safety shall always take precedence over device security. This means that any security mechanisms implemented should not unreasonably hinder access to the device during an emergency situation [2, 1]. Mechanisms should be devised that strike a good balance between device security and accessibility.

SR-1.5 **Security compliance with maintenance tasks:** Maintenance includes firmware updates (implemented as SiMS/SISC-core program-memory flashing), system-wide diagnostic checks, debugging mode of operation etc. Permission to execute such tasks is only granted to the medical technician (user role: "Technician") being responsible for the proper operation of the implant.

3 Automotive application

Innovation in cars is mainly driven by embedded software. To support such innovation, vehicles are being connected more-and-more, and thus becoming a mobile communication node in various directions. Examples of vehicle communication includes (see Figure 3.1) ECU-to-ECU[1], Car-to-Car, Car-to-Infrastructure, mobile phones, cloud services, open internet applications, Bluetooth and Wifi connections as well as diagnostic services.

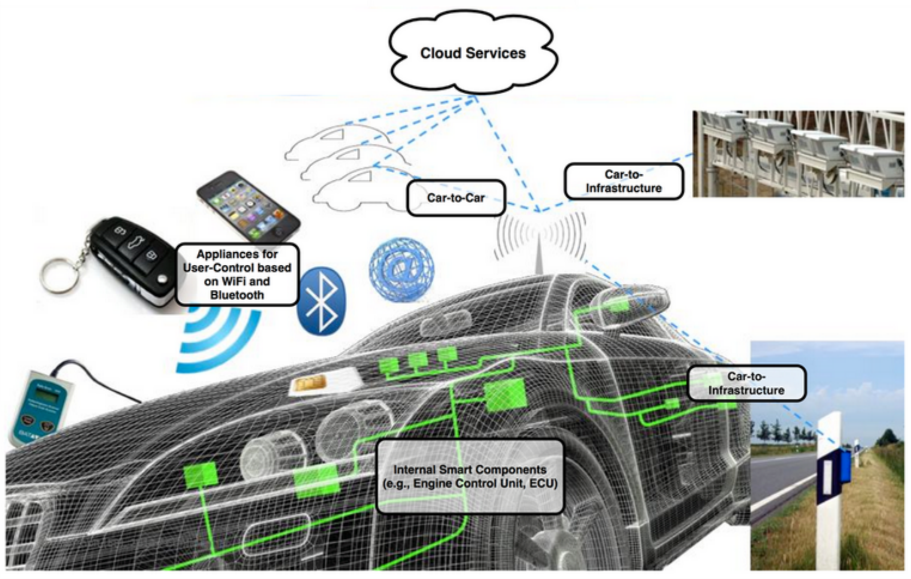
Cloud Services
Car-to-Car
Car-to-Infrastructure
Appliances for User-Control based on WiFi and Bluetooth
Car-to-Infrastructure
Internal Smart Components (e.g., Engine Control Unit, ECU)

Figure 3.1: Examples of vehicle communication

While existing connections to the outside world are usually used for applications in the entertainment domain, future applications will rely on in-

[1]ECU: Electronic Control Unit

formation from connected devices inside and outside the vehicle for safety critical use cases like assisted and autonomous driving or for example the exchange of traffic and hazard information between cars and roadside infrastructure.

Aspects of securing communication paths in the sense of classical IT-Security or data protection are already addressed in several initiatives like Car2X, Car2Car[2], EVITA[3] and AUTOSAR[4]. The protection of the actual ECU itself and its software on the other hand is not yet covered adequately.

Through SHARCS onboard and offboard vehicle communication will be enhanced with secure hardware and software to prevent attackers from manipulating functionality or gaining unauthorised access to those ECUs. Such protection is essential for the safety and security of passengers on the road.

Therefore, this application scenario is mainly aligned by the first SHARCS-framework operation model. In certain cases where hardware modifications are not an option software security mechanisms will be used as suggested by the second operational model (refer to Figure 3.2 for a summary of the SHARCS framework operation models).

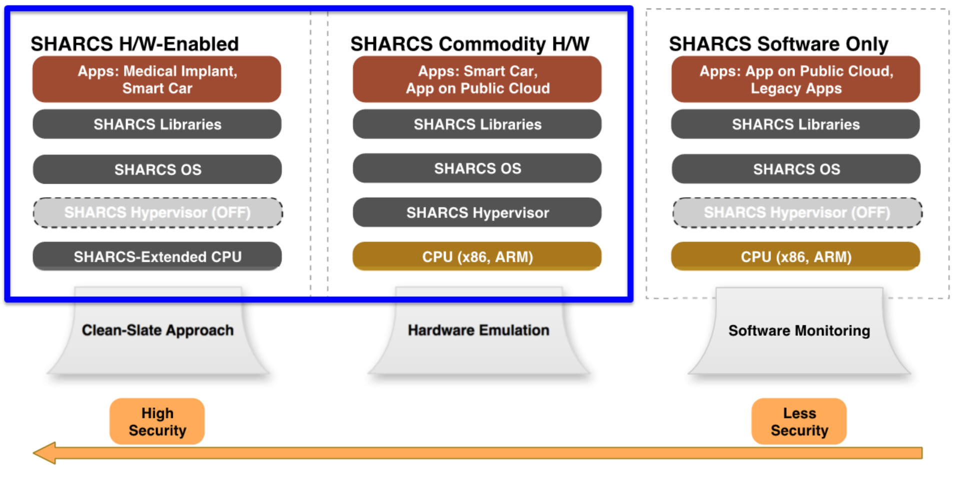

Figure 3.2: SHARCS-framework operational models, highlighting the areas relevant to the car application

3.1 System description

Modern premium cars have up to 80 Electronic Control Units ECUs. Figure 3.3 shows a selection of different ECUs mounted in a car. Some ECUs,

[2]http://www.car-to-car.org
[3]http://evita-project.org
[4]http://www.autosar.org

like the engine control and braking system, are essential in driving the car. Other systems such as the airbag are responsible for the safety of the driver. Window lift and Seat control are examples of ECUs controlling comfort functions, which are not necessarily needed to drive the car. The ECUs are interconnected to an on-board network by different automotive busses like CAN, Flexray or Ethernet. Furthermore a smart car may communicate with the outside world (e.g. Car-to-Car, Cloud service). Within SHARCS we would like to secure the complete system using a holistic approach.

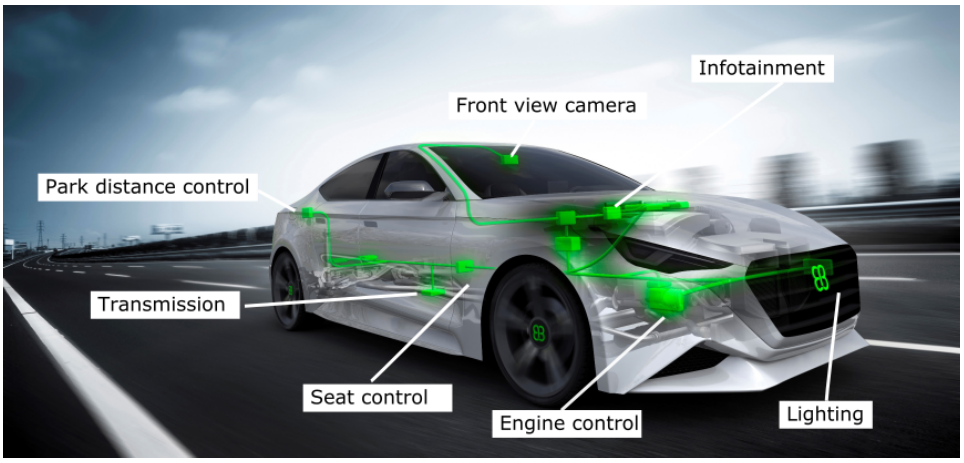

Figure 3.3: Automotive ECU examples

3.1.1 General description of a typical ECU

An Electronic Control Units ECU is an embedded system with a specific functionality in a car (e.g. engine control, brake). Figure 3.4 shows an abstract overview of an ECU. Depending on the ECU functionality there are different sensors(input) and actuators(output) connected. For example a Heating, Ventilation and Air Conditioning (HVAC) system needs a temperature sensor as an input. Depending on the measured and target temperature the system can control a heater. The ECUs are interconnected by different automotive buses like CAN, Flexray or Ethernet. It is therefore important to secure all ECUs regardless of their function. Otherwise it could be possible for a compromised ECU to gain access to others.

A typical automotive ECU consists, among others, of the following components:

- A processor core.
- Flash memory with a flash memory controller for persistent data storage.

- Memory Protection Unit (MPU).
- A clock control unit.
- An interrupt controller handling external and internal interrupts.
- Controllers for access to communication networks (e.g. CAN, Flexray, LIN, Automotive Ethernet).
- Sensor interface (e.g. Temperature, Speed, Camera, Radar).
- Interfaces for actuators (e.g. Actuators, Motors, Lamps, Relays).

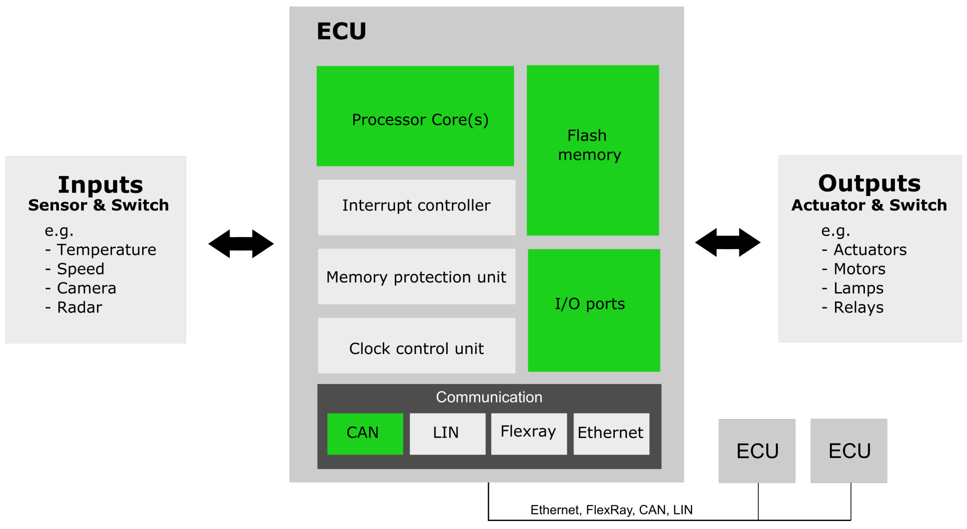

Figure 3.4: Typical ECU

In an ECU usually a software application based on the AUTOSAR software stack is executed on a specific automotive microcontroller (see Figure 3.7). The layered architecture ensures the decoupling of the functionality from the supporting hardware and software services. The AUTOSAR software stack is described in more detail in Chapter 3.1.4.

3.1.2 General description of a typical car network

All ECUs are interconnected to an on-board network by different automotive busses like CAN, Flexray or Ethernet. The on-board network architecture is different between every car manufacturer and even car model. A network instance is shown in Figure 3.5. All ECUs are usually combined into groups like for example Body Electronics (e.g. Window Lift, Lighting), Infotainment (e.g. Head Unit, Instrument Cluster), Chassis and Safety (e.g. Electric Power Steering, Airbag) and Powertrain (e.g. Engine control, transmission).

Communication between the different groups is possible over a gateway. It is therefore important to secure all ECUs as well as the communication between them regardless of their function. Otherwise it could be possible for a compromised ECU to gain access to others.

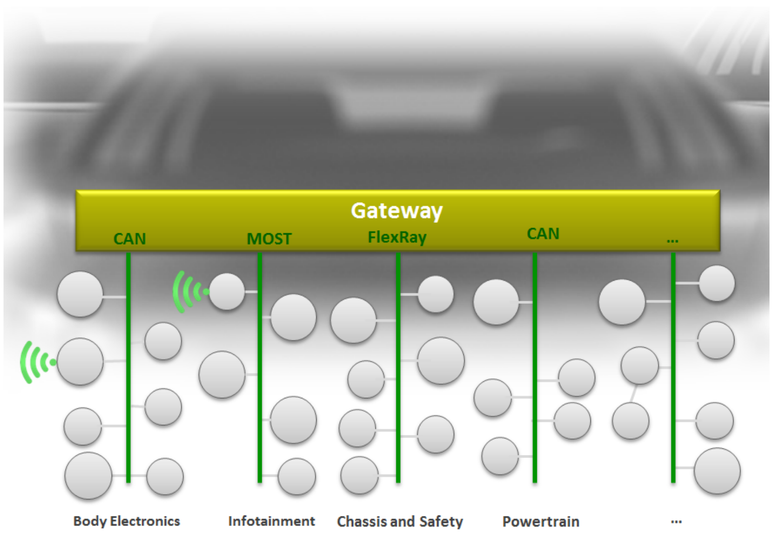

Figure 3.5: Automotive on-board network example

3.1.3 General description of a connected car

Modern cars are more and more connected to the outside world (Figure 3.6). They can communicate with other cars (Car-to-Car), with all kinds of infrastructure (Car-to-Infrastructure), with Cloud Services (e.g. real-time navigation or backup of settings) and with user appliances, such as smart-phones, which can control it remotely. Because of the wireless connectivity a possible vulnerability can be remotely exploited in a large number of vehicles. This must be prevented by security mechanisms against all hazards.

3.1.4 Autosar stack description

The AUTOSAR[5] (AUTomotive Open System ARchitecture) standard provides a layered software architecture and is a worldwide development partnership of vehicle manufacturers, suppliers and other companies from the electronics, semiconductor and software industry. The layered architecture ensures the decoupling of the functionality from the supporting hardware and software services. The following layers are implemented in a complete AUTOSAR software stack (see Figure 3.7):

[5]http://www.autosar.org/

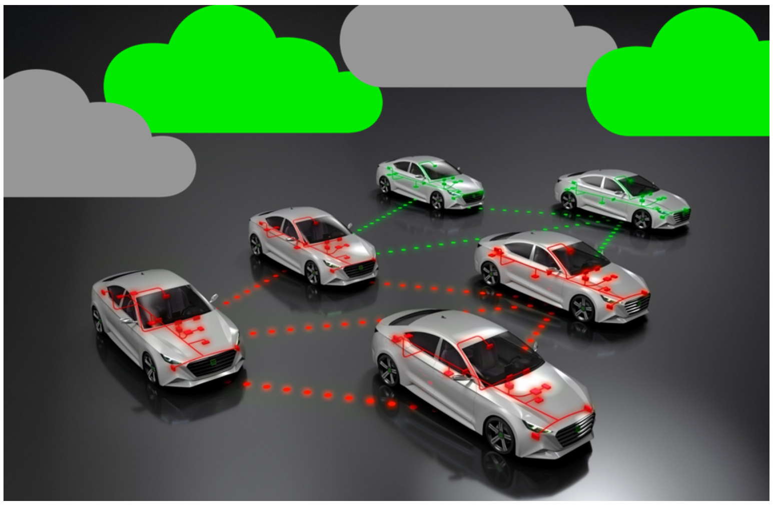

Figure 3.6: Connected-car example

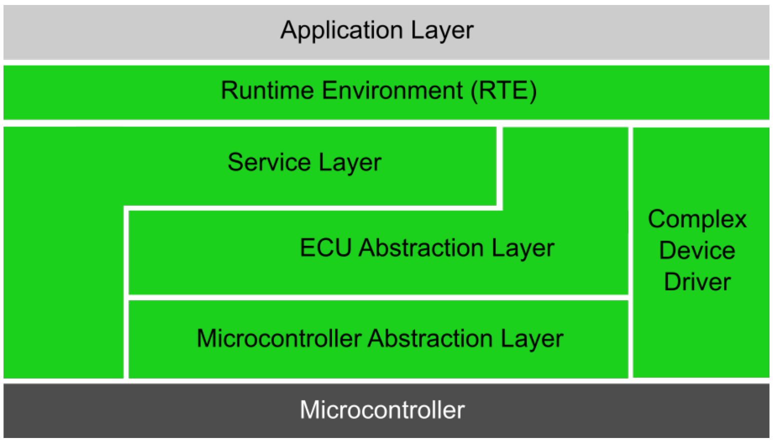

Figure 3.7: AUTOSAR layers

- The Autosar Stack is running on top of a microcontroller.
- The *Microcontroller Abstraction Layer* is the lowest software layer of the basic software. It contains internal drivers, which are software modules with direct access to the microcontroller internal peripherals and memory mapped external devices.

  Purpose: Make higher software layers independent of the microcontroller
- The *ECU Abstraction Layer* interfaces the drivers of the microcontroller Abstraction Layer. It also contains drivers for external devices like serial flash memories. It offers an API for access to peripherals and devices regardless of their location (microcontroller internal/external)

and their connection to the microcontroller (port pins, type of interface).

Purpose: Make higher software layers independent of ECU hardware layout, e.g. bus types, memory devices

- The *Complex Device Driver* implements complex sensor evaluation and actuator control with direct access to the MCU using specific interrupts and/or complex peripherals like PCP, TPU (e.g. Injection control, Electric valve control, Incremental position detection).

  Purpose: Fulfill the special functional and timing requirements for handling complex sensors and actuators

- The *Service Layer* is the highest layer of the Basic Software which also applies for its relevance for the application software: while access to I/O signals is covered by the ECU Abstraction Layer, the Services Layer offers: Operating system functionality, Vehicle network communication/management, Memory services (NVRAM management), Diagnostic Services (UDS, OBD), Mode management.

  Purpose: Provide basic services for application and basic software modules

- The *Runtime Environment (RTE)*is a layer providing communication services to the application software (AUTOSAR Software Components and/or AUTOSAR Sensor/Actuator components). Above the RTE the software architecture style changes from layered to component style. The AUTOSAR Software Components communicate with other components (inter and/or intra ECU) and/or services via the RTE.

  Purpose: Make AUTOSAR Software Components independent from the mapping to a specific ECU

- The *Application Layer* is a layer providing application software (AUTOSAR Software Components and/or AUTOSAR Sensor/Actuator components). Above the RTE the software architecture style changes from layered to component style. The AUTOSAR Software Components communicate with other components (inter and/or intra ECU) and/or services via the RTE.

  Purpose: Implement applications (runnables) that are executed by the RTE

In the SHARCS project we focus on the operating system and surrounding modules.

3.1.5 Application boundary - S/W

Elektrobit supplies complete software solutions to the automotive industry (see Figure 3.8). We provide the AUTOSAR software stack in source code and everything can be adapted/modified in the frame of the SHARCS project. More precisely EB provides:

- The AUTOSAR Basic Software (BSW) and Run Time Environment (RTE). This includes for example the operating system, bus communication and memory management.
- An AUTOSAR application consisting of software components (SW-C). In the frame of the SHARCS project this would be a simple demo application to show the effectiveness of the newly added security measures.

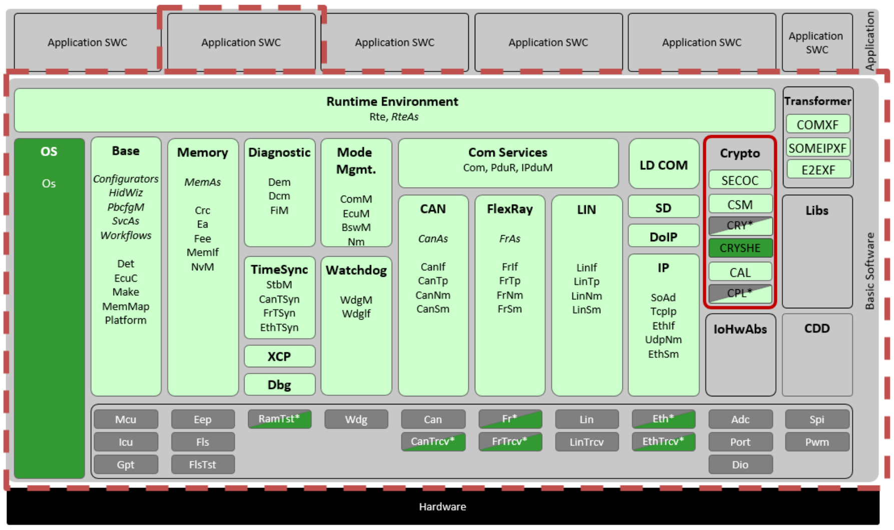

Figure 3.8: AUTOSAR software stack from Elektrobit

3.1.6 Application boundary - H/W

Elektrobits main focus is on software and the AUTOSAR stack implementation is suited to be used on off the shelf automotive hardware. However, hardware modifications of an automotive microntroller isn't feasible in the frame of the SHARCS project. As the Automotive Application is very security-critical we are also interested in the hardware security mechanisms and we are therefore using a prototyping platform instead.

3.1.7 User roles

The following user roles can be considered in an automotive application (see Table 3.1).

Table 3.1: Automotive-application user roles

Nr.	Roles	Permission level	Permissions
UR-2.1	ECU producer (Tier1)	Highest	Access and update the ECU at production
UR-2.2	Carmaker (OEM)		Access and update the ECU during the car lifecycle
UR-2.3	Garage		Access and update the ECU with OEM specific equipment
UR-2.4	Technical Inspection Authority/Police		Read out status information from the On-board diagnostics (OBD) interface
UR-2.5	End user (car owner/driver)	Lowest	No special access permission

3.1.8 Scope - existing security

Embedded security is not entirely new to the automotive domain. Elektrobit provides security mechanisms for more than 15 years to car manufacturers. For several typical automotive use cases partially standardised and many individual solutions exist and are used in vehicles already. The following security use cases are covered today:

- Authentication
- Signature
- Flash protection
- SW-Enabling (OEM-specific or according to HIS)
- Anti-theft mechanisms in SW
- Mileage protection
- Secure Onboard Communication
- Data protection
- Secure Hardware Extension (SHE)

- Hardware Security Module (HSM)
- Secure Boot
- Microkernel OS with memory and execution protection

Figure 3.9 shows an example setup with current security mechanisms in place. All of the solutions above usually rely on the use of cryptography. For the SHARCS project one shall assume that the existing mechanisms are (in general) suitable and working as intended.

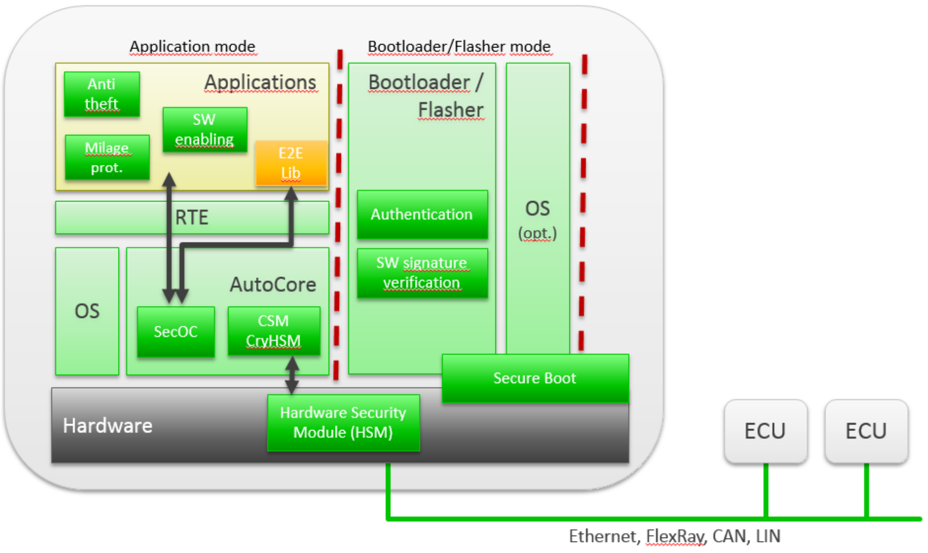

Figure 3.9: Example architecture with current security in place

3.2 Security evaluation

In spite of all established security mechanisms outlined in chapter 3.1.8, vulnerabilities might exist in a complex ECU, e.g. due to:

- Implementation errors
- Protocol flaws
- Unauthorised interaction between ECUs

In this chapter we describe the security threats and threat model in an automotive application.

3.2.1 Security threats

The following threats exist in the automotive application. Currently, confidentiality is not a primary security goal in the automotive industry, but that will change with highly connected use cases in the near future.

3.2.1.1 TH-2.1: Code/Data modification

Someone can insert code/data that alters the original behaviour of an ECU and e.g. endanger the safety of vehicles. A very prominent example is e.g. the Jeep attack[6].

- **Key security concepts / CIANA:** Integrity, (Availability), Non-Repudiation
- **Scenarios and involved components:**
  1. Unauthorised change of data in the memory (RAM, Flash or EEPROM) of the ECU that leads to different functionality (e.g. calibration data)
  2. Unauthorised change of executable code in the ECU that leads to different functionality

3.2.1.2 TH-2.2: Program flow modification

Someone can change the program flow and execute code which he is not allowed to. A typical attack on business cases would be to e.g. obtain optional software based functionality without paying for it (e.g. new navigation maps, advanced engine characteristics etc.).

- **Key security concepts / CIANA:** Integrity, (Availability), Authorisation
- **Scenarios and involved components:**
  1. Change of program counter
  2. Stack modification
  3. Privilege escalation by altering the processor state (user/privileged mode)
  4. MPU/MMU manipulation to execute none authorised memory regions

[6]http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

3.2.1.3 TH-2.3: Large scale exploit

Exploitation of a possible vulnerability in large scale (e.g. existing weak point can be remotely exploited in a large number of cars)

- **Key security concepts / CIANA:** Confidentiality, Integrity, Availability, Non-Repudiation, Authentication
- **Scenarios and involved components:**
  1. If an exploit for an implementation weakness is found a whole vehicle fleet with the same ECUs can be compromised (e.g. safety of the vehicle)

3.2.1.4 TH-2.4: Denial of service

A Denial of service attack could change the timing and execution of an ECU.

- **Key security concepts / CIANA:** Availability
- **Scenarios and involved components:**
  1. The ECU could not meet the timing requirements or even could hang up (e.g. brake system).

3.2.2 Threat model

The following assumptions regarding the thread model apply for the automotive use case:

AS-2.1 No physical modification of the ECU

AS-2.2 Physical access to ECU is possible (e.g. listen to externally accessible wired communication channels)

AS-2.3 The authentication credentials are unknown to an adversary

AS-2.4 The cryptographic cipher and security protocols are secure

AS-2.5 An attacker can send arbitrary messages to the ECU

AS-2.6 An attacker has access to an arbitrary number of ECUs (not necessarily mounted in a car)

3.3 Requirements

The automotive use case has the following application and security requirements.

3.3.1 Application requirements

AR-2.1 **Start-up time (150ms):** Main timing requirements for automotive controllers shall be met. This typically means that an ECU must be communication ready within 100 to 150 ms on the automotive bus after power on. At least diagnostics and network management capabilities must be provided to the entire network (*Hard constraint*).

AR-2.2 **Memory consumption security overhead (25%):** When providing additional security mechanisms in ECUs the increase in memory consumption shall not exceed 25% for RAM and/or ROM. (*Soft constraint*).

AR-2.3 **Execution time upper limit:** The different ECU applications have varying timing requirements. An engine control ECU for example is very timing critical and the newly developed security mechanisms must only have minimal influence on the performance. A window lift ECU in contrast will likely work properly even if the execution time is delayed. The timing impact from the new introduced security mechanisms should be therefore as minimal as possible to be usable in all the different ECU applications. (*Hard constraint*).

3.3.2 Security requirements

SR-2.1 **Message manipulation:** An attacker shall not be able to impersonate another sender of messages which are received by the controller on a communication bus in such a way that the controller executes code which the attacker provides.

SR-2.2 **Data flash manipulation:** If there is a data flash module on the controller which can be manipulated by an attacker, the attacker shall not be able to manipulate the data flash in such a way that the controller executes code which the attacker provides.

SR-2.3 **Single controller execution:** If the attacker is able to manipulate a single controller in such a way that the controller executes code which the attacker provides, the method used for this controller shall not be possible on a different controller running the same software stack.

SR-2.4 **Software module isolation:** If an attacker is able to modify the source code of one of the basic software modules or the application modules, the attacker shall not be able to obtain information about data in the other basic software modules and application modules.

4 Cloud application

OnApp builds and provides cloud software platform solutions based on multiple layers of cloud services. For SHARCS, we consider the OnApp Cloud Platform that is developed by OnApp. The end-to-end view of the deployment of OnApp Cloud from the repository to the customer Cloud is shown in Figure 4.1. The Cloud platform is used for virtualising hardware resources and presenting those resources to a set of end-users in the form of virtual machines (VMs). This helps data center owners to be able to achieve higher utilization of the resources and allows the management of the virtual machines presented by the hardware infrastructure to be handled by a company that specializes in cloud hosting. For end-users that just want web-hosting or other types of workload to run as a service, they don't have to be concerned with hardware maintenance and other issues that the data center owners will take care of.

End-users' VMs can run any type of workload that they would like as if it were a machine that they had full access to. They can choose to use an Operating System (OS) that is in the OnApp template repository or if they would prefer they can prepare a template from scratch. Normally a VM is

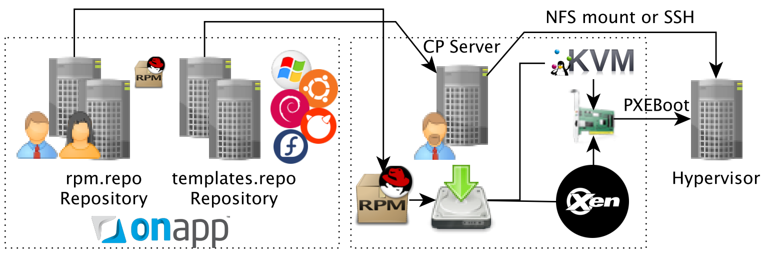

Figure 4.1: End-to-end view of the OnApp platform from the repository through to deployment

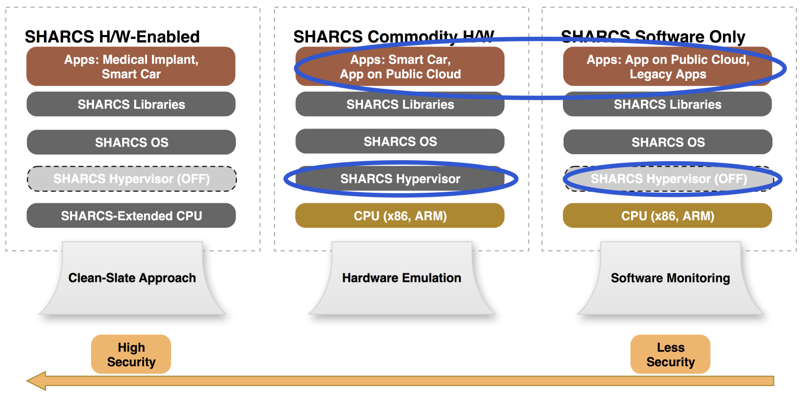

Figure 4.2: SHARCS-framework operational models, highlighting the areas relevant to Cloud Application

specialised to perform a single role and will be configured with software applications that allow it to function in that role. The workloads on a VM are normally considered to be the applications that are needed to perform the role with the rest of the system and the other tasks being considered as overhead. From the perspective of a hypervisor server on which these VMs are running the resources are shared as a black box, the workloads and overhead are considered together, when deciding how the resources should be shared between VMs.

The Cloud application relates more to the middle and right hand side of the SHARCS framework of operational models as highlighted in Figure 4.2. Depending on the Cloud operator, there may be the possibility of integrating changes in the Hypervisor platform. If there is little to no modification possible, we instead look at securing cloud applications that will be executed on un-trusted Cloud infrastructure.

The goal of securing this application is to

> **"distribute and run in a trusted manner, applications running in a VM on potentially un-trusted (cloud) systems that have limited ability to modify the hardware"**

To further specify this goal, the owner of VM$_1$ should be assured that a workload that they are running in VM$_1$ cannot have its data compromised or its performance affected beyond contracted Quality of Service (QoS) levels. These exploits could be indirectly caused by a secondary VM, VM$_2$ that may be misconfigured or compromised. In a Cloud environment we must consider that some legacy applications will not have the source code available whereas for some other applications the source code may be available. This applies for the host hypervisor platform, the guest operating system and the workloads. An example of a closed-source operating system is Microsoft

Windows 8.1 and CentOS 6.6 an example of an open-source operating system that is based on GNU/Linux.

The effort for securing the cloud application therefore will be two-fold;

1. Cloud Application Approach 1. Ensure the workload running in the cloud environment is secure - "Secure the execution of a workload"

2. Cloud Application Approach 2. Working from the bottom-up, assure that the cloud platform is secure; the hypervisors are secured, the virtual machine templates are secured and ensure that the workload is running in a secure environment - "Secure the cloud platform software"

Approach 1 will investigate making a workload hardened to vulnerabilities that are present in an untrusted environment. Approach 2 on the other hand will provide a complimentary approach to securing the cloud platform software from the hardware-level up to ensure that workloads not hardened as in the case of Approach 1 benefit from improved security. The full SHARCS end-to-end secured platform will combine and benefit from approaches 1 and 2.

4.1 System description

4.1.1 System boundary

The boundary of the system is considered to be the cloud deployment on the customer site. A minimal cloud deployment includes (as shown in Figure 4.3);

- Control Panel (CP) server that is the public facing server that manages the cloud deployment.
- A backup server is optional but highly recommended.
- One or more servers (two or more to have migration and other useful functions) that act as hypervisors for hosting virtual machines.

Normally a Storage Area Network (SAN) appliance is required for a cloud deployment to host the storage for the virtual machines. OnApp have a product, Integrated Storage[1], which means that for OnApp Cloud deployments an external SAN is not required. It is therefore assumed for this application that the storage is hosted on the hypervisors themselves and that Integrated Storage is used.

[1]http://www.onapp.com/storage

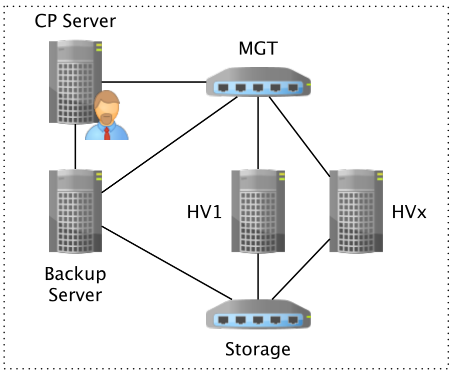

Figure 4.3: OnApp simplified Cloud deployment

Note

Note on cryptography vulnerabilities and threats: Many of the attacks and vulnerabilities for cloud-based computing systems come from attacks to the cryptographic elements in the platform, attacking weaknesses in the keys, fundamental security flaws, and flaws in the implementation of cryptographic mechanisms. Many projects cover the security of cryptographic components and as such they will not be directly addressed by SHARCS. For completeness some of the most prevalent vulnerabilities are captured in the Appendix (Section A.10).

Note

Note on hardware modification: Any changes to the underlying hardware will undermine many of the principles and techniques described. Given that data centers should normally have strict access controls to the hardware we will work on the basis that hardware cannot be modified and physical attacks are out of scope. Please also see the background on hardware in Appendix (see Section A.4).

4.1.2 User roles

There are multiple stakeholders involved in the cloud application that have different roles and permissions that add to the complexity of this application. Figure 4.4 shows a summary of the principle stakeholders involved and there is a brief description of the stakeholders contained in the list below. An end user (EU) will want to use the SHARCS platform to assure that the workload that they are running either directly or via an appliance owner (AO) is running in a trusted manner when there may be other workloads running in a shared tenancy (on the same hardware) environment.

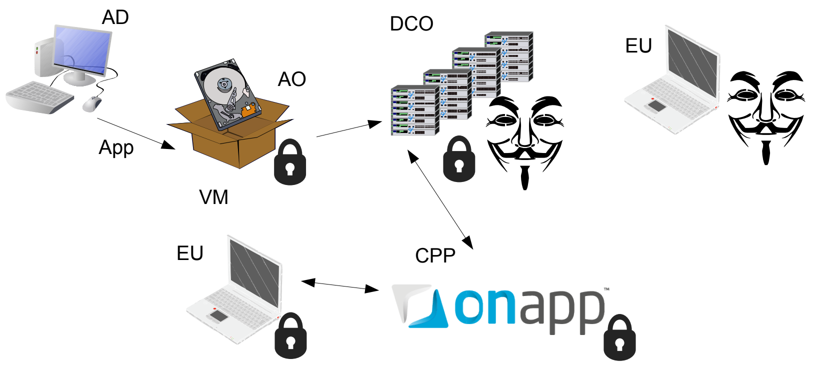

Figure 4.4: Securing the (un-)trusted cloud, scenario with stakeholders

UR-3.1 **The datacenter owner (DCO)** has a number of computing resources that are networked together in a physical location with external network connectivity.

UR-3.2 **The Cloud Platform Provider (CPP)** may be the same as a datacenter owner (DCO) or different but they provide a service to clients that expose the hardware resources in a controlled manner. They may use software platforms such as OnApp to expose these resources.

UR-3.3 **An appliance owner (AO)** is someone that uses the resources provided by the DCO directly or via a Cloud Platform Provider (CPP) to run a particular workload. In the case of a Software as a Service (SaaS) platform the AO will be created and managed by someone other than the EU.

UR-3.4 **Application Developer (AD)** creates an application that executes in a given environment.

UR-3.5 **An end user (EU)** uses or provides input/output to the application running on behalf of the AO.

At the core of the OnApp Cloud platform are servers that are configured as hypervisors, which host guest virtual machines. Within SHARCS we will consider both generic hypervisor and specific improvements to the Xen open-source hypervisor platform. For more details about the full list of services please see Section A.7.

A diagram based on the simplified Xen reference architecture[2] is shown in Figure 4.5. Hypervisor platforms can run directly on the hardware or

[2]https://blog.xenproject.org/2014/04/01/virtualization-on-arm-with-xen/

Table 4.1: Cloud-application user roles

ID	**Roles**	**Permission level**	**Permissions**
UR-3.1	Data center owner (DCO)	Highest for infrastructure	Power control of hardware. Ultimately responsible for physical infrastructure, connectivity and servicing
UR-3.2	Cloud platform provider (CPP)	Highest for administration of the service	Perform software updates of and manage the cloud platform. Handle resource allocation to users.
UR-3.3	Appliance owner (AO)	Highest for the set up and active servicing of the VM	Can change the workloads running. Can also update the running OS in the VM.
UR-3.4	Application developer (AD)	Low	Responsible for providing security updates for the upstream software packages
UR-3.5	End user (EU)	Highest for using the results of the workload.	Determines the way the workloads will be configured. Data control and privacy belongs to the end user unless delegated to another role.

as a guest to another operating system. For more information about the hypervisor platforms and what is commonly used by cloud providers please see Appendix (Section A.2). There are various privileged operations that are controlled through a specialised managed guest domain (that in Xen is known as Dom0) that controls access to the hardware.

The scenario investigates how cloud platforms (using OnApp Cloud platform as the basis for investigation) can be secured in such a way that a VM running a workload can be assured that it is running without data integrity being compromised and that the performance of the VM is not being affected due to over-provisioning etc, beyond a certain threshold. The hypervisor platform consists of a set of components that is depicted visually, in Figure 4.5. Each hypervisor server contains a HV layer above the hardware that is privileged and can only be accessed through the control domain. Other guest domains (VMs) can be created and managed by the control domain. Within each VM is an OS. On each OS the end-user may set up one or more workloads that are the set of applications needed to perform a particular role. The number of workloads and VMs supported by the platform is

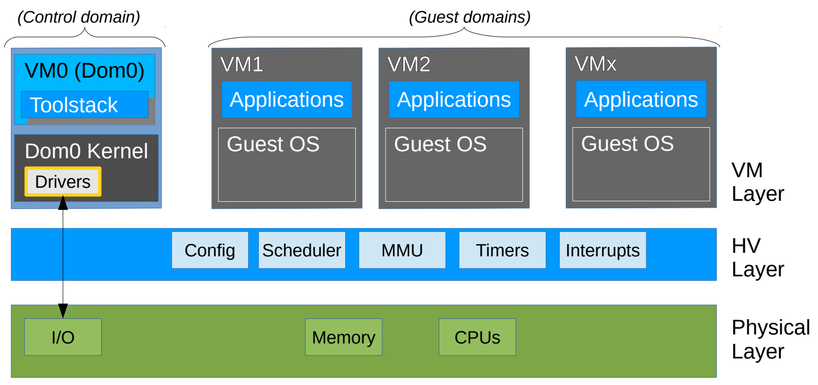

Figure 4.5: Simplified Xen architecture (based on Xenproject.org)

dependent on the configuration of the servers, the types of workloads, the OS type and the amount of physical resources available.

4.1.3 Scope - existing security

Cloud infrastructure as a service (IaaS) platforms leverage the existing security systems that are in place for traditional operating systems. Some of the principle security systems and existing techniques for security in the Cloud Application is described in the following list:

- Public-key based cryptography (SSH RSA 2048 bit) keys are inserted into the cloudboot image on the Control Panel (CP) server before the installation of the HVs

- For building packages we use the RPM Package Manager (RPM) build system

  MD5SUM check + digest of header.

  Packages can be signed with an OpenPGP key (not currently done)

- A recommended set of ports to be opened is provided online

- Heavily dependent on Xen / Kernel-based Virtual Machine (KVM) isolation mechanisms

  Libvirt/QEMU

- OnApp insert Xen Security Advisory (XSA) patches for hypervisor (HV) (Common Vulnerabilities and Exposure (CVE) DB exists)

- Update periodically (not systematically but using human intervention) the CentOS packages

  At the time of writing we have 240 packages in Xen and 377 for KVM cloudboot distributions

  Note: for Cloudboot we use Vault packages (e.g. locked down security patches are not back-ported)

- The Control Panel dashboard can enforce strong passwords and password rotation after a specified time

4.2 Security evaluation

In this section, based on the functionality and components described, we now evaluate the security of the cloud application. First, we describe the threats associated with a security breach (Section 4.2.1), as well as the threat model (Section 4.2.2) that specifies the assumptions under which we strive to guarantee security. Based on the security threats and threat model, we derive several attacks (included in Appendix - see Section A.5) which could potentially result in an impact for one or more of the stakeholders involved.

According to a report produced by Verizon[3] research in June 2015, ten CVEs account for 97% of the total exploits observed in 2014. The report also stated that 70% of attacks exploited known vulnerabilities that had a patch available.

In this particular application we want to secure a workload that is running within a VM to assure that the data integrity is secured and also to assure that it cannot run operations that affect the performance of other guests outside of the expected and contracted levels. Attacks that specifically target the utilisation of resources and integrity of individual workloads is considered in this application. Our focus is to protect workload integrity and also ensure that confidentiality of data between isolated resources is maintained unless explicitly authorised. Given that one of the main benefits of 'Cloud computing' is high availability of services we also want to assure availability if possible.

Taking into account the Xen reference model that was described earlier in Section 4.1 we now describe the attack surfaces present in a hypervisor platform, see Figure 4.6;

1. The control or management domain (Dom0 in Xen terminology). Any compromise of the control domain means that the interaction to the underlying hardware can be exploited.

[3] http://www.verizonenterprise.com/DBIR/2015/

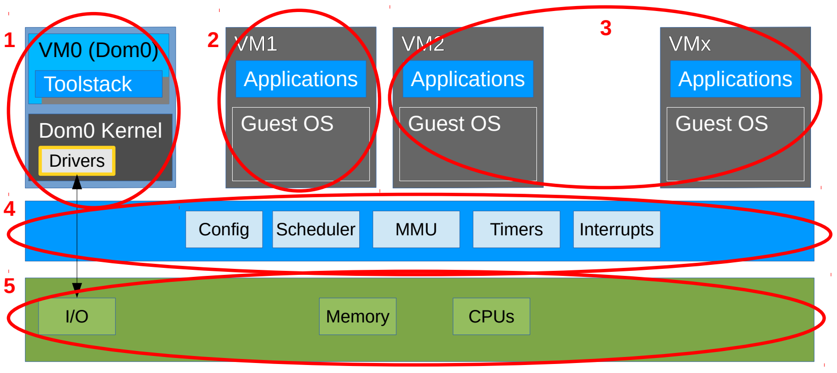

Figure 4.6: Simplified Xen architecture with reference to particular attack surfaces that are highlighted

2. A single VM can be attacked in many ways. The guest operating system may be compromised, the applications and services running on that VM may be compromised.

3. Side-channel attacks where VMs can attack other VMs through misconfigurations are not the focus of SHARCS as the attack surface is too large. Instead VM $\leftrightarrow$ HV security is considered and by working on end-to-end approaches it is expected that the number of possible side-channel attacks will be reduced.

4. The HV platform itself can be compromised and/or modified to expose security resources available in Hardware. In the scope of SHARCS we will be looking to modify hypervisor platforms in general (though for specific implementations we will focus on Xen type architectures given experience with the platform from the partners involved).

5. The underlying hardware in the Cloud application for SHARCS will be considered hard to modify. Certain mechanisms that can be used to secure particular pieces of hardware through software techniques will be considered for the hypervisor (area 4) and Dom0 (area 1).

Areas 1, 2 and 4; the control domain, individual VMs/workloads and the hypervisor platform will be the focus for the cloud application in SHARCS.

For more information about how the Control Panel and the hypervisors are set up please see the Appendix (A.6).

4.2.1 Security threats

The Cloud Security Alliance (CSA) and European Network and Information Security Agency (ENISA) have described a set of security threats that are relevant to this discussion. To not distract from the main flow of the text, the security threats are captured in the Appendix (see Section A.9). With reference to the earlier Figures 4.3 and 4.6 and the references captured in the Appendix, the security threats are captured below. The CIANA information partly uses information from the Cloud Security Alliance (CSA) and European Network and Information Security Agency (ENISA) sources included in the Appendix (see Section A.1).

4.2.1.1 TH-3.1: Unauthorised access to the cloud platform and/or VMs

Server/VM privilege escalation attacks (access to resources).

- **Key security concepts / CIANA:** Confidentiality, Integrity, Availability, Non-repudiation, Authentication
- **Scenarios and involved components:**
  1. VM escape to gain access to HV
  2. Exploit of a software vulnerability on the HV or CP that is remotely accessible or accessible from a VM
  3. Exploit of a misconfiguration in the network topology or set up that should limit access (Components: all)
  4. Exploit a vulnerability in application programming interfaces (APIs) to gain access (Components: all)
  5. Support user or administrator that abuses privilege - see RI-3.9
  6. Business practice or weakness in ACL system leaves users on the system record without revoking rights - see RI-3.7
  7. *Compromised security keys (Components: all)*
  8. *Unauthorised physical access to the system (Components: all)*

4.2.1.2 TH-3.2: Denial of service

- **Key security concepts / CIANA:** Availability
- **Scenarios and involved components:**
  1. Use of resources by a VM above the permitted levels agreed in Service Level Agreement (SLA)/contract

2. Exploit of vulnerability (including weakness in schedulers) (Components: HV)
3. Misconfiguration of resource sharing applied to schedulers (Components: HV)
4. Power control over VMs via RI-3.1
5. Insufficient resource limiting features available on a per user basis (architectural inadequacies)
6. Overcommit of resources by Cloud Service Provider (CSP)
7. Abuse of shared resources may result in deactivation for other shared tenants either internally or externally
8. Platform

4.2.1.3 TH-3.3: Modification of data

- **Key security concepts / CIANA:** Confidentiality, Integrity, Non-repudiation, Authentication
- **Scenarios and involved components:**
  1. Exploit vulnerabilities in shared Input / Output (I/O) virtualisation technologies
  2. Exploit vulnerability in ACL of memory regions
  3. Exploit vulnerabilities in drivers or firmware
  4. Modify data over shared I/O paths and/or compromise isolation mechanisms for HV
  5. Modify data in shared services (e.g. shared DB or file storage system)
  6. Misconfiguration or inconsistent view of resource boundaries
  7. *Modification of nearby data through hardware implementation flaws such as the RowHammer attack*

4.2.1.4 TH-3.4: Resource leakage

- **Key security concepts / CIANA:** Confidentiality, Integrity, Availability
- **Scenarios and involved components:**
  1. Misconfiguration or exploit of HV isolation properties
  2. Misconfiguration of VM post compromise of CP via 4.2.1.1
  3. Misconfiguration of schedulers and resource allocators
  4. Lack of platform and workload monitoring

4.2.1.5 TH-3.5: Breach or loss of data

- **Key security concepts / CIANA:** Confidentiality, Integrity, Availability, Non-repudiation, Authentication
- **Scenarios and involved components:**
  1. Misconfiguration or exploit of HV isolation properties
  2. Misconfiguration of VM post compromise of CP via 4.2.1.1
  3. Backdoors or other monitoring applications that have higher privilege levels than VMs
  4. Rogue application running in a VM
  5. Exploit of vulnerability in higher privilege applications or services running on CP or HV level
  6. Reading data on shared IO channels and/or other shared hardware resources
  7. Incorrect policies or techniques for deletion of data
  8. Backup and redundancy policy copies data to a device or system not covered by the security aspects
  9. Data migration or copying to an external system
  10. Exploit vulnerability in a shared application or library that is used for processing data
  11. Encryption keys for access of data corrupted or lost
  12. Reliability of underlying hardware
  13. *Top level keys and/or security certificates compromised*
  14. Insecure business practices

In addition to the security tasks present in the cloud platform there are a set of threats that are applicable when moving from a privately hosted workload to one that is hosted publicly. These include but are not limited to:

4.2.1.6 TH-3.6: Abuse and nefarious use of cloud computing

Attackers can then create workloads that target other platforms or internal systems. This can affect the reputation of the Cloud service provider and if not monitored, other platforms may disable the access to resources that this platform uses, therefore performing a denial of service for other users on the platform.

- **Key security concepts / CIANA:** Availability, Non-repudiation

- **Scenarios and involved components:**
  1. Weak or simple registration systems
  2. Lack of traceability for end users and their actions
  3. Lack of internal controls and/or monitoring
  4. Weak or insufficient logging mechanisms
  5. Modifiable logs
  6. Session riding
  7. Weak or unenforceable SLAs and contracts
  8. Administrators not reacting (or sufficiently quickly) to requests by law enforcement agencies

4.2.1.7 TH-3.7: Insufficient due diligence and shadow IT

If there is a flaw in the HV platform, or a misconfiguration of the system it is up to the CSP to detect and then to apply a patch in-time or to reconfigure the system. This relies on the Cloud Platform Provider (CPP) getting a software patch ready and the Cloud Provider (CP) and/or the DCO scheduling maintenance time.

- **Key security concepts / CIANA:** Confidentiality, Integrity
- **Scenarios and involved components:**
  1. Insecure devices and systems operating in the cloud infrastructure environment
  2. Weak and/or slow operational procedures for updating software platforms and responding to incidents
  3. Lack of compliance with GRC practices enforceable in the region that the data is hosted and processed
  4. Lack of regulation for CSPs
  5. Lack of accountability for CSPs
  6. Management and lifecycle of data
  7. Unclear roles split between CSP and end-users
  8. Weak or undefined roles and permissions in contracts and SLAs
  9. Insufficient and inconsistent reporting of incidents from CSPs to end-users
  10. Undefined procedures and actions for various levels of incidents

4.2.1.8 TH-3.8: External focused security systems.

Data center and cloud monitoring solutions are normally focused on the boundary point between the managed hardware and the Internet. Attacks are expected to be remote in nature and originate from external agents. This means that the monitoring systems and services may not be looking from attacks within the same data center or even on the same piece of hardware. Without appropriate active-monitoring, internal attacks may be hard or impossible to discover with potentially a larger lead time before the flaw is identified.

- **Key security concepts / CIANA:** Confidentiality, Integrity, Authentication
- **Scenarios and involved components:**
  1. A rogue VM could be set up to attack other VMs on the same HV and/or other VMs in the same data center
  2. Attacks can be launched on the hosting infrastructure from within the data center
  3. Internal network port scanning tools may find misconfigurations and vulnerabilities within the network
  4. Exploits could be used to create holes in the network from the inside
  5. Internal APIs may be weaker or less actively monitored
  6. Internal network operational procedures may differ from boundary points due to assumed trust
  7. Network monitoring hardware and software appliances may not be configured to monitor internal traffic

4.2.1.9 TH-3.9: Malicious insiders

Not only is the attack reward greater as mentioned in 4.2.1.6, leading to more potential external attackers, there are also other stakeholders associated with the cloud software platform provider, the data center owner and also third parties that are contracted by them. Support agents may have access to repair the software platform of the data center and may be in a position to compromise data of end-users due to the increased access rights used for maintaining and fixing the system. Often, the third parties are not communicated to the end-users. Aside from physical access to the hardware, support admins may have access to cloud workloads that they are unauthorised to and it may be difficult or impossible to track down any unauthorised access to the customer platforms. This kind of security threat has been demonstrated via the Edward Snowden release of sensitive, confidential data.

- **Key security concepts / CIANA:** Confidentiality, Integrity, Non-repudiation, Authentication
- **Scenarios and involved components:**
  1. Cloud platform developers include a backdoor or security vulnerability that can be exploited
  2. System stack developers may have left a vulnerability in their code
  3. Support or temporary keys not revoked after usage
  4. Ex-employee or employee with sufficient motivation may compromise the system
  5. Governmental or regional jurisdiction may mandate backdoors
  6. Hardware or low level firmware vulnerabilities built in
  7. Networking hardware and Internet traffic may be compromised, internally or at the boundary point
  8. Competitors may wish to determine operational differences and business practices
  9. *Security protocols purposefully broken or vulnerable to a particular set of attackers*
  10. *Master security keys or certificate authorities compromised*

4.2.2 Threat model

Research by various groups has different threat models for the cloud-application environment. For example, in CloudVisor [8] they take the threat model that the underlying hypervisor and management VM have been compromised and attach a lightweight security monitor beneath the hypervisor that encrypts and manages all access to I/O and shared memory resources. Other approaches such as HyperSafe [7] target improving the security of the hypervisor platform through control-flow integrity. In SHARCS, we take a holistic approach that will need to improve the security of the workload running in a VM on potentially untrusted hardware while at the same time improving the security primitives that are passed through to a VM and improving the security of the HV platform.

The following assumptions regarding the thread model apply for the cloud use case:

AS-3.1 In this scenario the RPM package is assumed to have arrived at the client with full integrity (please see Section A.8).

AS-3.2 Hardware is hard to attack: To limit the scope of this scenario we will assume that physical access to the hardware is only possible by the owner of the hardware and not by an attacker that can compromise the hardware. Physical modification of the hardware is therefore assumed to be impossible and not considered.

AS-3.3 From the perspective of the end-user, the cloud platform will be considered as potentially untrusted, unless trust can be attested.

AS-3.4 BIOS reconfiguration is only possible by the Cloud owner.

AS-3.5 A more generalised form of **AS-3.4** is that hardware in the infrastructure will remain in its set state unless exposed via API calls. We are assuming that attacks to exposed management ports and/or backdoors are not targeted.

AS-3.6 Password / credentials are maintained securely by individuals working on the system. We are assuming that social engineering techniques will not work and there is no password leakage.

AS-3.7 Supplementary packages will be required and/or patches to Xen and KVM unless the changes are pulled upstream.

AS-3.8 As described in Section 4.2, side channel attacks are not investigated as the configuration complexity is considered too great to address in SHARCS.

AS-3.9 End-users will have unrestricted access to the VM that they are paying for.

AS-3.10 All CSPs will provide a multi-tenant, shared access system unless otherwise specified.

4.3 Requirements

4.3.1 Application requirements

More details and background into the specific application requirements is captured in the Appendix (see Section A.11).

AR-3.1 **Performance of the system:** should not degrade more than 5% [soft requirement]

AR-3.2 **Selectable performance:** If performance is degraded beyond limit specified in **AR-3.1** then the security feature must be selectable with an impact when off of less than 5% [hard requirement]

AR-3.3 **Use regular hardware:** Hardware modifications are restricted to standard PCI Express plug in cards [hard requirement]

AR-3.4 **Backwards compatibility with existing infrastructure:** Any changes or modifications to the configuration of existing network hardware and the infrastructure must be backwards compatible [hard requirement]

AR-3.5 **Infrastructure configuration:** Configuration of network and infrastructure should not change [soft requirement]

AR-3.6 **Software in packaged form:** Software should be made available as installable packages to be included in beta platforms [soft requirement]

AR-3.7 **Software in packaged form:** Software must be available as a maintained package to be included in full OnApp platform [hard requirement]

AR-3.8 **Size of SHARCS software:** Software packages introduced by SHARCS additions should be no more than 30MB [soft requirement]

AR-3.9 **Size of SHARCS software:** Software packages introduced by SHARCS additions must not be more than 50MB [hard requirement]

AR-3.10 **Only established services:** Software services can only be enabled if they are well-established and tested [hard requirement]

AR-3.11 **Performance of additional services:** Performance impact of enabling services and the software configuration should not be greater than **[AR-C-1]** [soft requirement]

AR-3.12 **Packages require a full time maintainer:** For HV platform changes to be accepted in the OnApp production platform they must have been accepted upstream by the maintainers of the HV platforms [hard requirement]

AR-3.13 **Permanent data stored on CP:** All logging and permanent capturing of data must be done on the Control Panel server as the ramdisk images are loaded into RAM and are lost on reset [hard requirement]

AR-3.14 **Limited changes within a VM:** Customers may prepare their own VMs from templates. Templates therefore have very few modifications from the original OS. OnApp as a company policy do not place monitoring software or any other components other than those that are the minimal required for the template to function on a hypervisor [soft requirement]

As listed in Section 4.3.1 the hard constraints must be met regardless of any new security enhancements of SHARCS for the cloud application. The soft constraints can be violated to the point that they do not cause violation of the hard constraints in the final SHARCS-enabled cloud application.

4.3.2 Security requirements

SR-3.1 **End-user security and privacy:** The data in a VM that is used and generated by an end-user (user role: "End-user") is private and should not be accessible by other end-users or any other stakeholders unless authorised to do so.

SR-3.2 **Integrity of the platform and workloads:** The platform and workloads running in VMs must be resilient to attacks and degrade cleanly and in an expected manner. The integrity of the data of the data is more important and should not be lost or modified unless explicitly stated by the end-user (user role: "end-user") or by the Cloud service provider (user role: "Cloud service provider") if a contractual obligation has failed.

SR-3.3 **Availability of the platform:** The platform, unless it has a scheduled maintenance period that is organised by the CSP (user role: "Cloud service provider") should be available within the terms of the contractually bound SLA. The performance of the service must also not degrade beyond the contractual level as set out in the SLA.

SR-3.4 **All operations must be attributable to a user:** This necessitates that the users must be authenticated in the platform and that all operations that change the state of the platform are captured as being performed by one of the users.

SR-3.5 **All operations must be authenticated:** All changes to the running of the service including security audits and logging should be captured and viewable only by the stakeholders who are granted access to perform such operations.

In addition there are a number of GRC requirements. Depending on the region where the software will be run, different sets of requirements will need to be met. Some of the most common are detailed in the Appendix (see Section A.12).

Bibliography

[1] FDA. Content of premarket submissions for management of cybersecurity in medical devices guidance for industry and food and drug administration staff. http://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/UCM356190.pdf, October 2 2014.

[2] D. Halperin, T. S. Heydt-Benjamin, K. Fu, T. Kohno, and W. H. Maisel. Security and Privacy for Implantable Medical Devices. *IEEE Pervasive Computing*, 7:30–39, January 2008.

[3] ISO. Information technology – Security techniques – Entity authentication – Part 2: Mechanisms using symmetric encipherment algorithms, ISO/IEC 9798-2:2008. International Standard, 2nd ed., 1999.

[4] C. Strydis. *Universal Processor Architecture for Biomedical Implants: The SiMS Project*. PhD thesis, Delft University of Technology, Delft, Netherlands, March 2011.

[5] C. Strydis, R. M. Seepers, P. Peris-Lopez, D. Siskos, and I. Sourdis. A system architecture, processor, and communication protocol for secure implants. *ACM Trans. Archit. Code Optim.*, 10(4):57:1–57:23, Dec. 2013.

[6] M. van Dongen, A. Karapatis, L. Kros, O. Eelkman Rooda, R. Seepers, C. Strydis, C. De Zeeuw, F. Hoebeek, and W. Serdijn. An implementation of a wavelet-based seizure detection filter suitable for realtime closed-loop epileptic seizure suppression. In *Biomedical Circuits and Systems Conference (BioCAS), 2014 IEEE*, pages 504–507, Oct 2014.

[7] Z. Wang and X. Jiang. Hypersafe: A lightweight approach to provide lifetime hypervisor control-flow integrity. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 380–395. IEEE, 2010.

[8] F. Zhang, J. Chen, H. Chen, and B. Zang. Cloudvisor: Retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 203–216. ACM, 2011.

A
Appendix

In the following sections we include background information that is seen as relevant to the discussion of the Cloud application in D2.1. Each of the sub-sections are referenced in the main text as necessary. Effort has been undertaken to make these background items readable as stand-alone items but for the readers benefit it should be read in the context of the main text.

A.1 Cloud-application background

From a 2012 study of responses in 2011 by a large data center operator, Interxion, they found that security had the largest (45% of respondents) impact on cloud service uptake.

> ...[T]he biggest barriers to implementing cloud computing [was] a perceived lack of security and Service Level Agreements (SLAs) – *source Interxion 2012*[1]

In 2015 a white-paper[2] produced by the same company and IDG Connect stated that security was still the largest barrier (increased to 53%) for adoption in cloud computing uptake. The second barrier for migrating to cloud services was data protection and governance rules (41% of respondents).

A.2 Hypervisor types and usage by Cloud providers

Xen runs directly on the hardware and as such is considered a type-1 hypervisor platform. Xen then determines how resources should be partitioned between all the guest virtual machines and when the workloads should be

[1] http://www.interxion.com/globalassets/documents/whitepapers-and-pdfs/cloud/WP_CloudSurvey_en_0B.pdf
[2] www.interxion.com/IDGHybridIT

scheduled. Through this resource division and controlled access to privileged operations, Xen provides certain isolation guarantees that allow each virtual machine to appear like a full machine that should be isolated from all other systems running on the same hardware. Different public cloud providers use various hypervisor platforms and technologies. Amazon is the largest public cloud provider and utilises modified ports of earlier versions of Xen. Microsoft Azure utilises their Hyper-V[3] technology. Google compute engine uses the Linux powered KVM[4].

A.3 Common terms and groups related to Cloud security

There are many entities that have been involved in the creation of security practices related to the cloud. Some of have come and gone and others have become more widely-established than others. We now list (non-exhaustively) some of the most relevant terms that are linked to cloud security.

Computer systems incident report teams (CSIRTs) exist for a number of countries. A list capturing the incident report teams for many nations is maintained online by CERT[5]. A list of computing security resources can be found online[6] and includes but is not limited to;

CERT : The CERT[7] Division is an organization devoted to ensuring that appropriate technology and systems management practices are used to resist attacks on networked systems and to limit damage and ensure continuity of critical services in spite of successful attacks, accidents, or failures.

FIRST : The Forum of Incident Response and Security Teams (FIRST)[8] is a forum for communication of various incident report teams. It was founded in 1990, two years after CERT was created. A list containing the number of members for each nation that is represented by SHARCS is included in Table A.1.

CVSS : The Common Vulnerability Scoring System (CVSS)[9] provides an open framework for communicating the characteristics and impacts of IT vulnerabilities. Its quantitative model ensures repeatable accurate

[3]http://www.microsoft.com/hyperv/
[4]http://www.linux-kvm.org/
[5]http://www.cert.org/incident-management/national-csirts/national-csirts.cfm
[6]http://www.cyberdegrees.org/resources/the-big-list/
[7]http://www.cert.org/faq/index.cfm
[8]http://www.first.org/about
[9]http://www.first.org/cvss

measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the scores. Thus, CVSS is well suited as a standard measurement system for industries, organizations, and governments that need accurate and consistent vulnerability impact scores. CVSS is managed by FIRST.

CSA : CSA[10], is a non-profit organization focused on best practices for security assurance within Cloud computing and education on using the Cloud to help secure all other forms of computing. The CSA have produced a detailed guidance document for various aspects of security in the cloud, which at the time of writing, the latest version is 3.0[11].

CVE : CVE[12] is a dictionary of publicly known information security vulnerabilities and exposures. CVEs common identifiers enable data exchange between security products and provide a baseline index point for evaluating coverage of tools and services.

US-CERT : The Department of Homeland Security's United States Computer Emergency Readiness Team (US-CERT)[13] leads efforts to improve the Nation's cybersecurity posture, coordinate cyber information sharing, and proactively manage cyber risks to the Nation while protecting the constitutional rights of Americans. US-CERT strives to be a trusted global leader in cybersecurity-collaborative, agile, and responsive in a dynamic and complex environment.

NVD : National Vulnerability Database (NVD)[14] is the U.S. government repository of standards based vulnerability management data represented using the Security Content Automation Protocol (SCAP). This data enables automation of vulnerability management, security measurement, and compliance. NVD includes databases of security checklists, security related software flaws, misconfigurations, product names, and impact metrics. NVD maintains a CVSS score for all CVE vulnerabilities.

CERIAS : The Center for Education and Research in Information Assurance and Security (CERIAS) is currently viewed as one of the worlds leading centers for research and education in areas of information security that are crucial to the protection of critical computing and communication infrastructure[15].

[10] https://cloudsecurityalliance.org/
[11] https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf
[12] https://cve.mitre.org/
[13] https://www.us-cert.gov/ncas/alerts
[14] https://nvd.nist.gov/
[15] http://www.cerias.purdue.edu/site/tools_and_resources/

OWASP : The Open Web Application Security Project (OWASP) is a 501(c)(3) worldwide not-for-profit charitable organization focused on improving the security of software. Our mission is to make software security visible, so that individuals and organizations worldwide can make informed decisions about true software security risks[16]. OWASP did maintain a list of security risks in the cloud in the now discontinued Cloud 10 project[17].

CVE Details : CVE Details[18] is a site that takes NVD Extensible Markup Language (XML) feeds and provides a graphical display of the data. In addition to the NVD data it also takes data from additional sources from exploit-db[19], vendor supplied data and metasploit[20] modules.

Exploit-DB : Exploit-DB[21] is a website that provides source code for testing CVE exploits that is maintained by a security training company, Offensive Security.

STRIDE : Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service, Elevation of Privilege (STRIDE) threat model is a mnemonic developed by Microsoft that indicates key threat categories, useful for identifying threats[22]. For each of the processes and components in the system the threats should be identified, collected and assessed.

CIANA : CIANA is another taxonomy used for categorising types of threats that originated from the key concepts of CIA; Confidentiality, Integrity and Availability. It is widely used by security researchers to identify the type of threats and vulnerabilities. Confidentiality, Integrity, Availability, Non-repudiation, Authentication

DREAD : Damage, Reproducibility, Exploitability, Affected users + Discoverability (DREAD), developed and previously used by Microsoft is used once threats have been identified for rating the security risks. The system was too subjective, with the implementation of the scoring system being the main issue and phased out of use by Microsoft in 2008.

CSIG : In February 2013, the EC set up the cloud select industry group (CSIG) subgroup on SLA C-SIG-SLA to work on aspects of standardisation of SLAs in the Cloud industry. On 26th June 2014 CSIG produced a version of its guidelines, referencing in turn more standards, into the

[16] https://www.owasp.org/
[17] https://www.owasp.org/images/4/47/Cloud-Top10-Security-Risks.pdf
[18] http://www.cvedetails.com/
[19] http://www.exploit-db.com/
[20] http://www.metasploit.com/
[21] https://www.exploit-db.com
[22] https://msdn.microsoft.com/en-us/library/ee823878%28v=cs.20%29.aspx

Table A.1: Number of FIRST Team members by countries that SHARCS partners are located

Country	Part. short name	No. of members
Greece	FORTH	1
Netherlands	VUA, NEU	9
Sweden	CTH	5
Germany	TUBS, EBA	23
United Kingdom	ONAPP	16
Israel	IBM	3

standardisation of SLAs[23]. It provides contributions to the EC and also provides its recommendations to the ISO/IEC 19086 project.

ENISA : "The ENISA is a centre of network and information security expertise for the EU, its member states, the private sector and Europe's citizens. ENISA works with these groups to develop advice and recommendations on good practice in information security." ENISA produced a report in December 2012 detailing the risks that it had identified that are captured in Section 4.2.

Other threat modeling systems : There are a variety of other threat modeling systems that have been developed. OWASP maintains a list of alternative threat modeling systems on their website[24].

A.4 Cloud-hardware assumptions

Hardware in data centers is normally standardised. We assume therefore that no changes are possible for the existing hardware. To ensure that we do not build a system that relies on hardware modifications that will not be usable by data center providers we should assume that the hardware available will be from standard vendors such as Dell, HP etc. Custom hardware and the addition of modules or add-in cards that are not available as standard will severely reduce the impact and exploitability of results. Not all servers have PCI-Express slots and/or USB slots that the data center owners will be happy to populate with additional hardware. Trusted platform modules (TPMs) available on motherboards and/or processor extensions that are available to Intel Xeon and AMD Opteron chips should be considered. This corresponds with Objective 2 of SHARCS- *"leverage features present in todays processors and hardware"*.

[23]http://ec.europa.eu/information_society/newsroom/cf/dae/document.cfm?action=display&doc_id=6138

[24]https://www.owasp.org/index.php/Threat_Risk_Modeling

A.5 Cloud-application vulnerabilities

The following vulnerabilities are from academia.edu[25] - via (cloudcomputing-news[26])

Cloud-computing vulnerabilities from Academia.edu

- Session Riding: Attacker steals a cookie and then can perform operations in the name of the user. Cross-site request forgery (CSRF) may also be used for sending authenticated requests to arbitrary web sites.
- Virtual Machine Escape: Exploit a Hypervisor remotely using an HV vulnerability. A threat may also use a virtual machine escape to leave the sandbox environment and gain access to the rest of the HV and VMs available on that system.
- Reliability and Availability of Service: Need to take into consideration that services are not 100% guaranteed and may have some down-time, relative to the SLAs.
- Insecure Cryptography: Servers do not have access to randomisation sources available on desktop machines and so have less entropy available. As stated in A-C-7 we assume that cryptography is sufficiently secure.
- Data Protection and Portability: When migrating to/from a CSP the data will have to be deleted to a sufficient level. If a CSP goes out of business the assets may be sold on and could potentially contain confidential data.
- Cloud Service Provider (CSP) Lock-In: The ability to move between providers reduces the risk that having lock-in entails.
- Internet dependency: The infrastructure of the Internet relevant to a particular provider may also be affected by temporary failures, reducing availability to services.

According to the Verizon[27] report the usage of the most exploited CVEs accounted for 97% of exploits and can be seen in Figure A.1. With the exception of two flaws reported since 2012 the majority are from flaws predating 2003.

Examples of two major vulnerabilities that caused CSPs to restart a proportion of VM instances in late 2014 and early 2015 were Shellshock and GHOST.

[25] http://www.academia.edu/4877213/Seven_Deadly_Threats_and_Vulnerabilities_in_Cloud_Computing

[26] http://www.cloudcomputing-news.net/news/2014/nov/21/top-cloud-computing-threats-and-vulnerabilities-enterprise-environment/

[27] http://www.verizonenterprise.com/DBIR/2015/

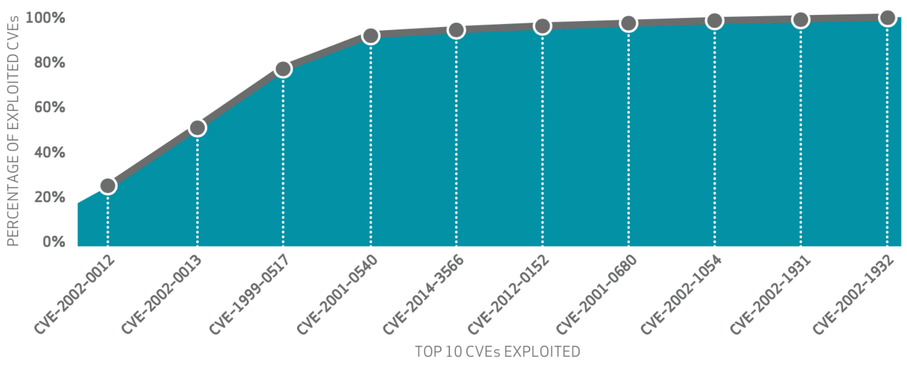

Figure A.1: Cumulative percentage of exploited vulnerabilities by top 10 CVEs – *source Verizon*

Shellshock : Shellshock[28] is a collection of widely publicised security vulnerabilities of the Borne Again Shell (BASH) that is used in many UNIX systems. The initial security flaw was assigned CVE-2014-6271 and was patched. Additional flaws were found shortly after using similar techniques and subsequent patches released.

GHOST : The GHOST vulnerability CVE-2015-0235[29] affects the hostname function of particular versions of glibc (2.2 and other 2.x versions before 2.18) and if exploited allows remote code execution.

A.6 Configuration of Control Panel and hypervisors

In the cloud application the data-flow can be configured in almost any manner applicable to applications. What is useful to capture are the interfaces of the workload with a given platform. Cloud applications are typically installed on a remote OS environment that is either pre-prepared (by the CSP) or configured by the end-user. This OS will be the primary interface for a (virtual machine (VM)/ resource) container and will then interact with the resources presented by the HV and have a remotely accessible Internet Protocol (IP) address. This connection will be made via a console interface, a Virtual Network Computing (VNC) connection or via Secure Shell (SSH) or similar secured connections. The primary interaction will then be via this interface until the operating system is configured to allow more ports to be opened, depending on the required configuration. Management of the VM resources is normally provided via a web-based user-interface (dashboard) that allows an authorised and authenticated user to view the resources associated with their account. There may be multiple VMs linked to an account

[28] https://en.wikipedia.org/wiki/Shellshock_(software_bug)
[29] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-0235

of varying sizes that may be configured into groups or separated. Also tied in with this system is the billing system that will normally be managed by the Cloud or data center operator that then bills for this use of resources at a level of granularity that is defined in the SLA. Normally a user will pay for the size of the resources that they want at a minimum level and it is up to the datacenter owner (DCO) and cloud manager to decide how those resources will be placed. Depending on SLAs and configuration the resources may in certain cases be over-committed or there may be guarantees of certain dedicated access in which case over-commit is disabled. Configuration options allow hints be provided to the HV platform to decide what quantity of resources should be allocated to which VMs at a certain time that act in a time-share manner access to the resources. When a user deletes a VM these resources then return to the available resource pool for other users to acquire those resources. Usually CPPs provide limited options for how workloads should be placed (e.g. no two servers on the same physical machine) but these don't take into account other users who are on the platform. If isolation properties are well enforced then a user should not be able to affect the running of another VM on the same physical machine through either direct or indirect attacks.

Administrative actions by Control Panel The HVs in a cloud are configured by the Control Panel (CP). The type of HV that is used on a server will be decided by the Cloud Service Provider (CSP). Depending on the configuration of the overall Cloud there may be some settings that then propagate to an individual HV. For instance on the OnApp platform the HV can be configured to use KVM, Xen for HV-types and then baremetal, smart or virtual to configure the type of server they will be. The CP manages the configuration of the VMs and the rest of the platform. The SSH keys for the HVs are stored on the CP as is the database and the dashboard. Any compromise to the CP will mean that the HVs can be easily compromised. If Cloudboot is used (a type of PXEBoot system) then the HV image can be modified on the CP, which is equivalent of modifying the image on a compromised HV disk drive (just a different source is used). Additionally any hardware that is set up to use the same VLANs and is in the same network if compromised could potentially become a rogue CP server that acts as a DHCP server and serves the requests of the HVs. This leads to the situation where a server that is not in the original cloud configuration or a compromised backup server could take over as a rogue CP server, creating the possibility for a Man-In-The-Middle (MITM) type attack. The CP server may for convenience also have the IPMI commands stored that detail how to control the power of the servers.

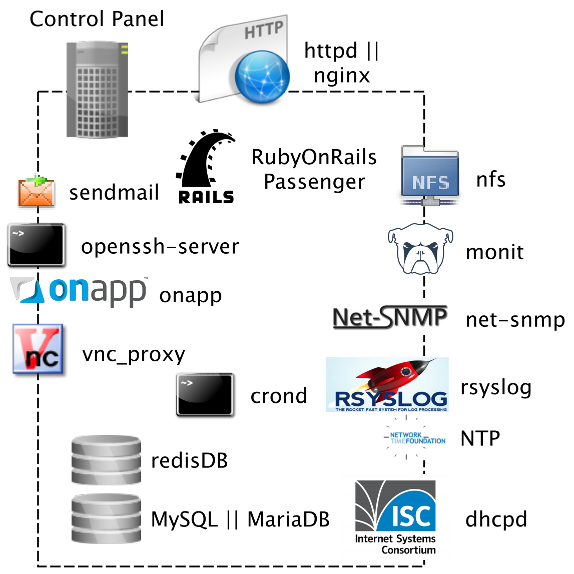

Figure A.2: The main services and components enabled by default on a CentOS 6 base system

A.7 Default OnApp components status

Table A.2 shows the changes to the default services and components in CentOS that are used for the Cloudboot image. The CP can be set up anyway that a customer would like but the services captured in the CP row show the services that are required (additionally the components are also shown). The services that are enabled on the CP server at run level 3 (default run-level) are; acpid, auditd, crond, dhcpd, haldaemon, httpd, libvirt-guests, messagebus, monit, mysqld, netfs, network, nfs, nfslock, ntpd, onapp, portreserve, rabbitmq-server, redis, rpcbind, rpcgssd, rsyslogd, sendmail, snmpd, sshd, udev-post, xinetd. The ports that OnApp needs to be opened can be seen on the web-site[30].

Both the CP and Cloudboot HVs are customised to utilise different services and components. A diagrammatic representation of some of the key services for the CP is shown in Figure A.2. Similarly for the HV a diagram of the main Internet related services that are enabled is shown in Figure A.3.

A.8 RPM preparation and deployment

Early in the SHARCS project the Cloud application was split into two areas; the development and deployment system and secondly, securing the

[30]https://docs.onapp.com/display/MISC/Required+Ports

Table A.2: The components and services that are needed on CentOS by OnApp. Services highlighted in **bold** are (en/)disabled by OnApp

Component	**HV type**	**Services**	**Status**
CentOS 5	Xen	**syslog**, xend, xendomains	on
		mcstrans	off
CentOS 6	KVM	**openibd, opensmd**	on
		ksm, ksmtuned	off
CentOS 6	KVM/Xen	**rsyslog**	on
		mcelogd, cupsd	off
CentOS 5/6	KVM/Xen	SANController, crond, groupmon, haldaemon, ip6tables, iptables, irqbalance, iscsi, iscsid, libvirt-guests, libvirtd, lm_sensors, lvm2-monitor, mdmonitor, messagebus, netfs, network, nfslock, ntpd, portmap, rpcgssd, snmptrapd, sshd, **ntpd, smartd, snmpd, tgtd, storageAPI**	on
		avahi-daemon, sysstat	off
CP	N/A	(auditd, blk-availability, **crond**, dhcpd, haldaemon, (**httpd** || nginx), iptables, iptables6, iscsi, iscsid, libvirt-guests, lvm2-monitor, mdmonitor, messagebus, monit, **mysqld** || mariadb), netfs, network, nfs, nfslock, ntpd, **onapp**, portreserve, postfix, **rabbitmq-server**, **redis**, rpcbind, rpcgssd, rsyslog, sshd, udev-post, xinetd	on
CP components	N/A	onapp-cp-redis, rails pkgs, onapp-cp-vnc-proxy, onapp-cp-rabbitmq, snmptrap, onapp-cp-monit	N/A

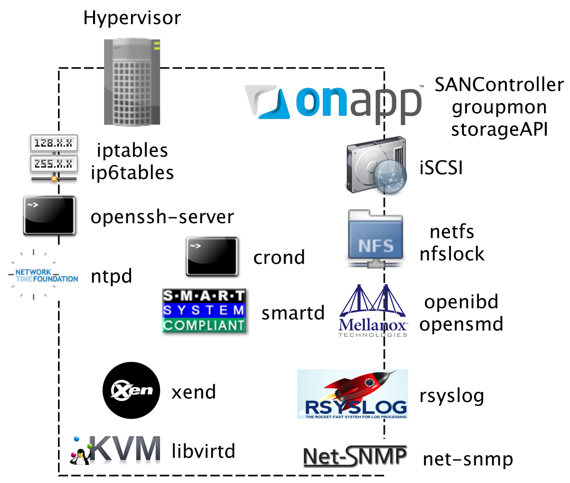

Figure A.3: The main services and components enabled by default on a Hypervisor

workload operation within the Cloud framework. The development and deployment part was discussed between partners as being an area of possible work that reflects a common need between all of the applications. The requirements are to have a secure build mechanism and then to ensure that the packages produced are securely transferred to the customer sites. Some changes to improve the build process and increase the security to that of good-practice will be under-taken. These include signing of the RPMs to ensure that the identity of the source is known and that users have accessible mechanisms to check that the packages have not been modified en-route. Rather than considering this a separate approach, the trusted boot execution will be investigated as part of the main approach. We want to ensure that the workloads running on the platform are secure, maintain integrity of the SHARCS hardened workloads and ensure that workloads are more difficult to compromise and if compromised the impact is limited. For the cloud application we will assume that the packages as prepared by OnApp have reached the customer site free of any modifications.

A.9 Cloud security-evaluation background

From Cloud Security Alliance (CSA) Cloud Computing vulnerability incidents report in 2012[31] they analysed a set of news articles for cloud computing via Google search engine and then highlighted cloud vulnerabilities

[31]https://cloudsecurityalliance.org/download/cloud-computing-vulnerability-incidents-a-statistical-overview/

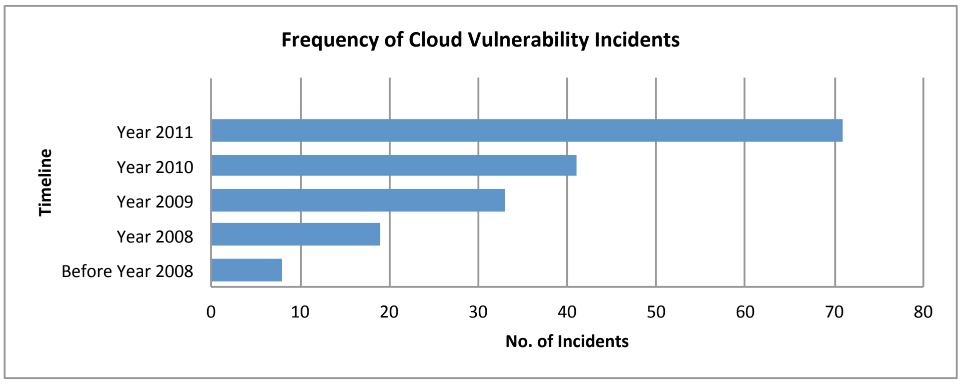

Figure A.4: Frequency of Cloud Vulnerability Incidents up to end of 2011 (title corrected) - source CSA

to highlight interesting trends in the first 5 years of cloud computing, see Figure A.4.

The CSA Top Threats as categorised in 2009[32] are shown in Table A.3 with the expanded list continuing into Table A.4 that included a number of categories to accommodate the other types of incidents.

Table A.3: Overview of CSA Top Threats v1.0

No.	CSA Top Threat
1	Abuse and Nefarious Use of Cloud Computing
2	Insecure Interfaces and APIs
3	Malicious Insiders
4	Shared Technology Issues
5	Data Loss or Leakage
6	Account or Service Hijacking
7	Unknown Risk Profile

Table A.4: Description of new threats uncovered by CSA

No. (New Threat)	Cause of Vulnerability
8	Hardware Failure
9	Natural Disasters
10	Closure of Cloud Service
11	Cloud-related Malware
12	Inadequate Infrastructure Design and Planning

[32] https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf

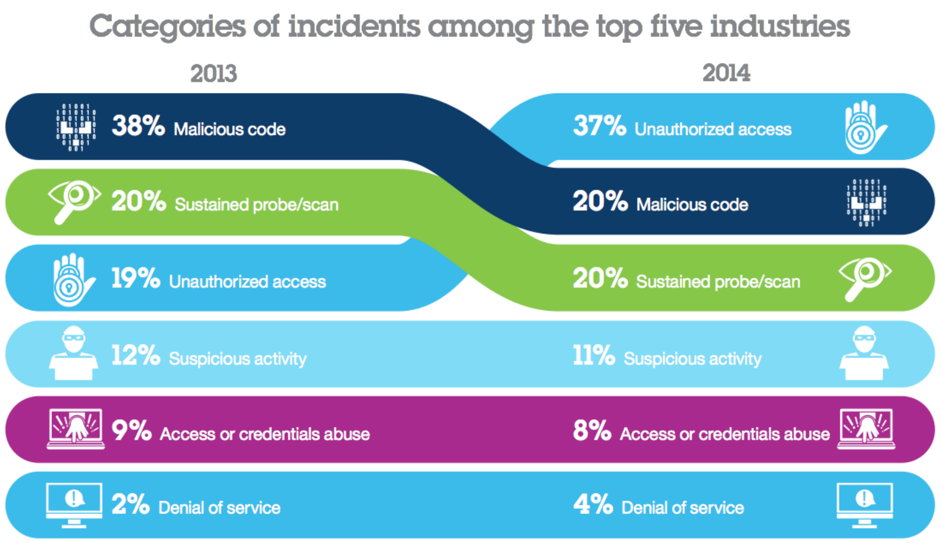

Figure A.5: Changing percentage of security incidents between 2014-2015 - source IBM 2015 Cyber Security Intelligence Index

The types of threats change with time as highlighted in Figure A.5[33].

The priority is data integrity and assuring that data from one VM cannot leak into another VM in any way. The next most important risk is to protect against changes in the results of another VM, even if the resources of the VM are not directly accessible (indirect attacks).

The 5 cloud risks as described by InfoWorld InfoWorld[34].

Shared access : The same computing resources; Central Processing Unit (CPU), storage, memory, namespace, [network] and physical building [may be shared by multiple tenants]. If there is a compromise it may allow an attacker to assume the identity of another client and cause a data breach. Software that has not been previously cleaned or is leaked to another customer could lead to the leakage of records. IP address re-use might also be possible with a client assuming an address that was previously associated with another customer leading to further potential identity based attacks.

Virtual exploits : The virtualisation platform adds additional, unique threats to a standard system including potential vulnerabilities to server host only, guest to guest, host to guest and guest to host attacks.

[33] http://www-03.ibm.com/security/data-breach/2015-cyber-security-index.html

[34] http://www.infoworld.com/article/2614369/security/the-5-cloud-risks-you-have-to-stop-ignoring.html

Authentication, authorisation and accounting (AAA) control : Not only does the Authentication, authorisation and accounting (AAA) system of the cloud software platform provider lead to potential security vulnerabilities, the process that a CSP chooses may lead to other vulnerabilities. This may include the processes involved for clearing out stale accounts, the use of shared-spaces, the re-use of private keys to secure multiple virtual machines or the associated resources. This also entails the physical location and sovereignty of the data that is stored. The life-cycle and handling of storage and information once a VM and/or customer is no longer using the system may also provide further attack vectors.

Availability : Redundancy and fault tolerance are often stated as benefits of cloud computing but the extra redundancy and copies may lead to insecure backups and other potential vulnerabilities. The liability for a CSP to the data held on the cloud may vary by location and agreement and so although a backup scheme may be elected by an end-user, unless it is frequently tested and/or an external backup system used this could lead to an issue in the availability of the workloads.

Ownership : When uploading and using public cloud services, there are often clauses in the SLAs and contracts that detail that the cloud provider becomes a part or sole owner of the data that is uploaded. This means that unless data is explicitly protected, the CSP may search or mine the data for revenue.

CSA top 9 threats to Cloud Computing From the Cloud Security Alliance (CSA)'s report on the top 9 threats to Cloud computing produced in 2013[35] the additional risks below are captured. The risks are ranked in order of severity.

Note

The risk values have been taken from the report's graphs and converted into a scale of 0-10 with 10 being the far right or far top of the graph, 5 being the mid-point lines and 0 the axes.

R-CSA-1 : Data breaches
Access to data from either a single or multiple tenants in a shared hosting space
Actual Risk 7. Perceived Risk 7.
CIANA: Confidentiality. STRIDE: Information Disclosure

[35] https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf

R-CSA-2 : Data Loss
Data that is not backed up and only preserved on the Cloud infrastructure is at risk. Similarly data that is only backed up to the cloud also has a risk. This also includes access to data, which may be affected due to data corruption or the loss of an encryption key.
Actual Risk 6. Perceived Risk 7.
CIANA: Availability, Non-Repudiation. STRIDE: Repudiation, Denial of Service

R-CSA-3 : Account Hijacking
Attack methods such as phishing, fraud and exploitation of software vulnerabilities. Impact is made larger due to credential re-use.
Actual Risk 8. Perceived Risk 7.2
CIANA: Authenticity, Integrity, Confidentiality, Non-repudiation, Availability. STRIDE: Tampering with Data, Repudiation, Information Disclosure, Elevation of Privilege, Spoofing Identity.

R-CSA-4 : Insecure APIs
Software APIs are normally made available by cloud providers to allow services to be exposed. Incorrectly configured services may be abused. Other services may be built on insecure APIs further increasing the impact.
Actual Risk 5. Perceived Risk 6.
CIANA: Authenticity, Integrity, Confidentiality. STRIDE: Tampering with Data, Repudiation, Information Disclosure, Elevation of Privilege

R-CSA-5 : Denial of Service (DoS) attacks impact the end-users' ability to access services by slowing down to intolerable levels.
Actual Risk 5.5. Perceived Risk 8.7.
CIANA: Availability. STRIDE: Denial of Service

R-CSA-6: Malicious Insiders A malicious insider can intentionally modify the network and/or system to negatively impact the confidentiality, integrity and availability of the organisations information or information systems.
Actual Risk 4.1. Perceived Risk 6.9.
CIANA: N/A. STRIDE: Spoofing, Tampering, Information Disclosure

R-CSA-7 : Abuse of Cloud services
Utilise the large computing power available to run services that would otherwise be impossible for a smaller organisation to perform.
Actual Risk N/A. Perceived Risk N/A.
CIANA: N/A. STRIDE: N/A

R-CSA-8 : Insufficient Due Diligence
Companies may rush to cloud computing without understanding the

full impacts and implications of moving their services to the Cloud.
Actual Risk 8.2. Perceived Risk 1.5.
CIANA: N/A. STRIDE: All

R-CSA-9 : Shared Technology Issues
The inherent scalability that makes Cloud computing attractive, means that resources that were not meant to be grouped together may be linked, presenting weaknesses in a multi-tenant architecture.
Actual Risk 2. Perceived Risk 3.7.
CIANA: N/A. STRIDE: Information Disclosure, Elevation of Privilege.

In an interview with computerweekly[36] it is recorded that the CTO of Azure stated at the annual IP Expo 2014 in London - "This is important because there is no cloud without trust." The top ten threats reported by him have been covered previously.

Shared tech vulnerabilities : See [R-CSA-9]

Insufficient due-diligence and shadow IT : See [R-CSA-8]

Abuse of cloud services : See [R-CSA-7]

Malicious insiders : See [R-CSA-6]

Denial of service : See [R-CSA-5]

Insecure interfaces and APIs : See [R-CSA-4]

Unauthorised access to an enterprise user's cloud account : Similar to [R-CSA-3]

Data loss : See [R-CSA-2]

Data breach : See [R-CSA-1]

Self-awareness or AI : This relates more to processes related to cloud.

ENISA suggested cloud risks The European Network and Information Security Agency (ENISA) also produced a set of risks that is captured in their December 2012 report[37], highlighting the top ten estimated risks for cloud computing. The authors stated that the "top risks have turned out to be more or less unchanged from the 2009 Cloud Risk Assessment."

[36] `http://www.computerweekly.com/news/2240232396/How-to-mitigate-top-ten-public-cloud-security-risks-Azure-CTO-Mark-Russinovich`
[37] `https://resilience.enisa.europa.eu/cloud-security-and-resilience/publications/cloud-computing-benefits-risks-and-recommendations-for-information-security`

R-ENISA-1 : Loss of governance due to ceding control to the Cloud Service Provider (CSP). "This also includes compliance risks."
Probability: Very High. Impact: Very High. Risk: Very High

R-ENISA-2 : Lock-in. Due to the lack of a converged standard there are many services, tools and processes used by each Cloud Platform Provider (CPP) and CSP. This can make it difficult to migrate between services.
Probability: High. Impact: Medium. Risk: High

R-ENISA-3 : Isolation failure. If complete isolation is not guaranteed then resources from other VMs may be compromised via the HV.
Probability: High. Impact: High. Risk: High

R-ENISA-4 : Management interface compromise.
Probability: Medium. Impact: Very high. Risk. High

R-ENISA-5 : Data protection
Probability: High. Impact: High. Risk: High.

R-ENISA-6 : Insecure or incomplete data deletion
Probability: Medium. Impact: Very high. Risk: High.

R-ENISA-7 : Malicious insider
Probability: Medium. Impact: Very High. Risk: High

R-ENISA-8 : Customers' security expectations

R-ENISA-9 : Availability Chain

R-ENISA-10 : Compromise of Service Engine
Probability: Low. Impact: Very high. Risk: High.

ENISA has continued this research and in February 2015 it reported[38] the top 8 risks, which are mainly the same as the 2012 risk list but include Compliance risks and removes the last three risks ([R-ENISA-8] to [R-ENISA-10]). The top risks are selected from a set of risks that have been categorised. In the most recent research there are 35 categories that have been identified.

A.10 Cryptographic exploits

The focus of SHARCS is not on cryptographic weaknesses and vulnerabilities. There are many projects whose focus is on ensuring secure protocol development and more secure initiation for sharing secrets and keys. Many

[38] http://www.clubcloudcomputing.com/2015/05/top-8-cloud-risks-according-to-enisa/

of the attacks in recent years on IT infrastructure at the Enterprise level and for Cloud security have targeted vulnerabilities in insecure implementations of the protocols for particular platforms. In SHARCS we advocate the use of encryption for end-to-end communication within and outside of the platforms. We now list of the more well publicised security vulnerabilities in cryptographic implementations in recent years.

Heartbleed : The Heartbleed[39] bug is a serious vulnerability in an implementation of the OpenSSL cryptographic software library.

POODLE : Padding Oracle On Downgraded Legacy[40] (POODLE) is an attack whereby Secure Socket Layer (SSL)v3 is allowed on an affected server instead of Transport-Layer Security (TLS) thus allowing a man-in-the-middle attack to decrypt content transferred on an SSLv3 connection.

Weak Diffie-Hellman / Logjam : Logjam[41] is a vulnerability that affects the TLS protocol and is described in various security sites including CSA[42]. The vulnerability can degrade a secure Diffie-Hellman 2048-bit algorithm to a lower level of encryption.

SMACK : State Machine AttaCKs[43] are a set of state machine related vulnerabilities in TLS implementations. FREAK is one such attack against SSL/TLS. A FREAK[44] attack is possible when a vulnerable browser connects to a susceptible web server - a server that accepts "export-grade" encryption.

A.11 Cloud application – application-requirements background

Performance of a single HV should not degrade more than X% (where X is suggested qualitatively to be five) due to additional security features. This figure is not based on any market studies and the actual value that customers are willing to accept may be far greater. Important performance figures for hosted Cloud providers are currently perceived performance; providers

[39] http://heartbleed.com/
[40] https://blogs.oracle.com/security/entry/information_about_ssl_poodle_vulnerability
[41] https://weakdh.org/
[42] https://blog.cloudsecurityalliance.org/2015/06/01/ciphercloud-risk-lab-details-logjam-tls-vulnerability-and-other-diffie-hellman-weakness/
[43] http://smacktls.com/
[44] https://freakattack.com/

though are increasingly using third party benchmark suites such as cloudbench[45], UnixBench[46] and others to compare performance. Benchmarks to be useful and realistic, run a given workload on one or more providers and compare the time taken to complete certain operations and measure the efficiency of selected performance attributes. For the Integrated Storage platform (OnApp's proprietary distributed storage system) a performance attribute that customers tend to compare is the level of Input/Output oPerations per seconds (IOPs). I/O on a virtualised system tends to provide the largest bottleneck for performance. IOPs can be measured using applications such as IOMeter[47]. Any security features that reduce the IOPs by up to 5% will likely be acceptable by customers. If there is a selectable set of security features that can be selectively (en/)disabled then the performance loss could be a tradeoff that customers could determine. It is likely if a particular set of security features has a noticeable impact on the performance of a hosted cloud system (e.g. more than the speculative 5% figure) that the features will need to be selectable and possibly disabled by default. Any security option that is selectable can have a greater impact on performance as long as it doesn't affect the performance when de-selected. The cloud application also has some constraints in that it is very hard to modify hardware on a customer site. The hardware that is provided may be refreshed periodically but there is no ability to change or modify the hardware from an end-user perspective. Data center owners will tend to have a maintenance contract with one or more systems vendors and will not be willing to use time and effort to modify the hardware that comes in. This means that it is very unlikely that data center owners will install add-in cards that are not provided by the manufacturer. The general infrastructure and network configuration should not require changes other than what would be expected of a standard data center. This means that the data center topology should not be changed and that the security features should be built with the assumption that they need to maintain backwards compatibility with the existing infrastructure and network topology. This expands upon the condition that no hardware should be changed to include the configuration of the system. For easier use in deployment and evaluation all SHARCS software packages that will be installed for the cloud application must be in the form of an installable package. Any software that is to be included in the core/storage product for OnApp platform must be in the form of an RPM package that should be available from upstream CentOS. This poses the additional requirement that a package must have a maintainer, which may or may not be possible in the timeframe of the SHARCS project. For development and analysis it is possible to temporarily include a SHARCS

[45] https://github.com/nephoscale/cloudbench
[46] https://github.com/kdlucas/byte-unixbench
[47] http://sourceforge.net/projects/iometer/

RPM that is included in the build system and will then be incorporated for beta versions. There must also be no software license restrictions for including code and its dependencies. Nothing should be included that requires a restricted package (e.g. shouldn't rely on non-export grade cryptography protocols). Adding new package dependencies is not always possible. We have a set of packages available for Xen and KVM and try to limit the size of the Cloudboot image so that it fits in Random-Access Memory (RAM). If the package or packages set is/are less than 20MB there will likely be no issues as long as there are no conflicts with the existing package set. The total size of the SHARCS package and the additional dependencies should not be more than 30MB but must not be more than 50MB (this can be verified by producing an image without the SHARCS package and then comparing against one with it and the dependencies included). Running services in the background on the start up of HVs and the Control Panel (CP) is possible but we try to limit these to known and well established services (e.g. contained in most, popular Linux distributions). It is unlikely that we would be able to add a SHARCS service as a start at boot time to anything other than a beta platform. We will be able to turn on services that are included but not enabled by default as long as the performance constraints described previously are maintained. For a list of services enabled by OnApp components please see the Table A.2 in the Appendix Section A.7. Changes to the HV platform must have been pushed upstream before they will be accepted in the production software platform but for the beta this constraint is relaxed.

A.12 Governance, risk management and compliance background

A.12.0.1 Governance, risk management and compliance

In the next list we will partly describe governance, risk management and compliance relevant to data centers. It is up to an individual data center provider to comply to the regulations of the location that they operate in. For the cloud application we must ensure that the methods and techniques do not break compliance. Improving the security mechanisms should improve the situation rather than be a hindrance for adoption of the cloud. If the SHARCS suite addresses some or all of the issues set out in the compliance sections then this will help to promote adoption of SHARCS. As a requirement though we must not modify any system in such a way to detriment the ability to comply with the regulations. KPMG has also produced a report aimed more at CEOs and CIOs of the compliance relevant to cloud

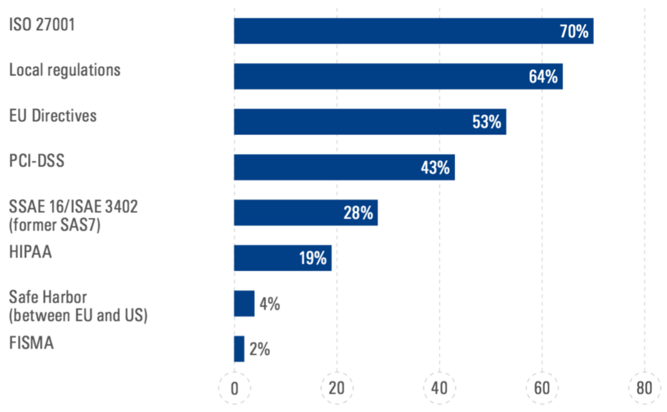

Figure A.6: The most important regulations to comply with based on interviews with companies – *source KPMG*

computing[48]. In their 2014 report they listed what regulations company owners felt most important to comply with and is captured in Figure A.6.

CR-C-1 : The Health Insurance Portability and Accountability Act (HIPAA)[49] of 1996 required the Secretary of the U.S. Department of Health and Human Services (HHS) to develop regulations protecting the privacy and security of certain health information. Part 160[50] and Part 164[51] contain the legal clauses relevant. A checklist of requirements can be found on the web also[52].

> Data centers need to adhere to the administrative, physical and technical safeguards and standards set forth by the HITECH[53] act of 2009 to be HIPAA compliant – *source datacenterknowledge*[54].

This is applicable for U.S. data centers that will host medical data.

[48] https://www.kpmg.com/HU/en/IssuesAndInsights/ArticlesPublications/Documents/20140718-IT-Security-Compliance-for-Cloud-Service-Providers-m.pdf

[49] http://www.hhs.gov/ocr/privacy/hipaa/understanding/srsummary.html

[50] http://www.gpo.gov/fdsys/pkg/CFR-2007-title45-vol1/content-detail.html

[51] http://www.gpo.gov/fdsys/pkg/CFR-2007-title45-vol1/content-detail.html

[52] http://www.ihs.gov/hipaa/documents/IHS_HIPAA_Security_Checklist.pdf

[53] http://www.hhs.gov/ocr/privacy/hipaa/administrative/enforcementrule/hitechenforcementifr.html

[54] http://www.datacenterknowledge.com/archives/2012/06/29/hipaa-compliant-data-centers/

CR-C-2 : ISO 27001[55].

> ISO/IEC 27001:2013 specifies the requirements for establishing, implementing, maintaining and continually improving an information security management system within the context of the organization. It also includes requirements for the assessment and treatment of information security risks tailored to the needs of the organization. The requirements set out in ISO/IEC 27001:2013 are generic and are intended to be applicable to all organizations, regardless of type, size or nature.

Organisations that meet the requirements can apply for a certificate of compliance.

CR-C-3 : ISO 27002[56].

> ISO/IEC 27002:2013 gives guidelines for organizational information security standards and information security management practices including the selection, implementation and management of controls taking into consideration the organization's information security risk environment(s).

Organisations that meet the requirements can apply for a certificate of compliance.

CR-C-4 : UK Data Protection Act[57] of 1998 covers accessibility and protection of user data.

CR-C-5 : EU General Data Protection Regulation (GDPR)[58] will impact any organisation that gathers, processes or stores personal data.

CR-C-6 : PCI-DSS v3.0. The Payment Card Industry Data Security Standard is a proprietary standard for handling payment information. For information relevant to CSPs there has been a supplementary guidance report[59].

CR-C-7 : CSA-STAR[60] is produced by CSA. It "is the industry's most powerful program for security assurance in the cloud. STAR encompasses

[55] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=54534

[56] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=54533

[57] http://www.legislation.gov.uk/ukpga/1998/29/contents

[58] http://www.itgovernance.co.uk/data-protection-dpa-and-eu-data-protection-regulation.aspx

[59] https://www.pcisecuritystandards.org/pdfs/PCI_DSS_v2_Cloud_Guidelines.pdf

[60] https://cloudsecurityalliance.org/star/

key principles of transparency, rigorous auditing, harmonization of standards, with continuous monitoring also available in late 2015." It is a form of industry self-regulation.

CR-C-8 : SSAE 16/ISAE 3402. Statement on Standards for Attestation Engagements (SSAE) is a way for assuring customers that use a data center that the processes and techniques are in line with what is expected[61] if the data center is hosting services relevant to customer financial reporting. ISAE 3402 is an assurance standard for service organisations that provides information for auditors relevant to understanding if effective controls are in place for managing financial reporting[62].

CR-C-9 : Directive 95/46/EC[63]. This is currently the latest directive produced by the EC in relation to protection of data[64]. It is likely to be superseded by the release of the General Data Protection Regulation due at the end of 2015. Safe Harbor[65] is a streamlined process for U.S. companies to comply with this directive.

[61] http://www.datacenterknowledge.com/archives/2011/09/27/why-data-centers-need-ssae-16/

[62] http://www.ifac.org/system/files/downloads/b014-2010-iaasb-handbook-isae-3402.pdf

[63] http://eur-lex.europa.eu/legal-content/en/ALL/?uri=CELEX:31995L0046

[64] http://ec.europa.eu/justice/data-protection/index_en.htm

[65] http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32000D0520:EN:HTML