SANCUS: Multi-layers Vulnerability Management Framework for Cloud-native 5G networks

CHARILAOS ZARAKOVITIS, National Center For Scientific Research "Demokritos", Greece
DIMITRIOS KLONIDIS, Ubitech Limited, Cyprus
ZUJANY SALAZAR, Montimage EURL, France
ANNA PRUDNIKOVA, Secura BV, Netherlands
ARASH BOZORGCHENANI, University Of Lancaster, United Kingdom
QIANG NI, University Of Lancaster, United Kingdom
CHARALAMBOS KLITIS, EBOS Technologies Limited, Cyprus
GEORGE GUIRGIS, EBOS Technologies Limited, Cyprus
ANA CAVALLI, Montimage EURL, France
NICHOLAS SGOUROS, Eight Bells LTD, Cyprus
EFTYCHIA MAKRI, Centre for Research and Technology Hellas/ Information Technologies Institute, Greece
ANTONIOS LALAS, Centre for Research and Technology Hellas/ Information Technologies Institute, Greece
KONSTANTINOS VOTIS, Centre for Research and Technology Hellas/ Information Technologies Institute, Greece
GEORGE AMPONIS, K3Y, Bulgaria
WISSAM MALLOULI, Montimage EURL, France

Abstract: Security, Trust and Reliability are crucial issues in mobile 5G networks from both hardware and software perspectives. These issues are of significant importance when considering implementations over distributed environments, i.e., corporate Cloud environment over massively virtualized infrastructures as envisioned in the 5G service provision paradigm. The SANCUS[1] solution intends providing a modular framework integrating different engines in order to enable next-generation 5G system networks to perform automated and intelligent analysis of their firmware images at massive scale, as well as the validation of applications and services. SANCUS also proposes a proactive risk assessment of network applications and services by means of maximising the overall system resilience in terms of security, privacy and reliability. This paper presents an overview of the SANCUS architecture in its current release as well as the pilots use cases that will be demonstrated at the end of the project and used for validating the concepts.

Additional Key Words and Phrases: 5G networks, Security, Trust, Reliability, Risk Detection and Mitigation

ACM Reference Format:
Charilaos Zarakovitis, Dimitrios Klonidis, Zujany Salazar, Anna Prudnikova, Arash Bozorgchenani, Qiang Ni, Charalambos Klitis, George Guirgis, Ana Cavalli, Nicholas Sgouros, Eftychia Makri, Antonios Lalas, Konstantinos Votis, George Amponis, and Wissam Mallouli. 2021. SANCUS: Multi-layers Vulnerability Management Framework for Cloud-native 5G networks. In *The 16th International*

[1]H2020 SANCUS project was started on September 1st, 2020 and lasts 3 years. More details can be found in: https://www.sancus-project.eu/

Manuscript submitted to ACM

Conference on Availability, Reliability and Security (ARES 2021), August 17–20, 2021, Vienna, Austria. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3465481.3470092

1 INTRODUCTION

Security-Trust-Reliability are crucial issues for hardware and application software installations, in particular considering implementations over heterogeneous distributed environments, as is the case for corporate Cloud environments deployed over massively virtualised infrastructures as envisioned in the 5G service provision paradigm [7], [1]. These environments require new tools and strategies, which enable next-generation ICT system networks to perform automated and intelligent analysis of their firmware images [5], and enabling proactive risk assessment of network applications and services by means of maximising the overall system resilience in joint terms of security, privacy and reliability [8]. Moreover, these environments are implemented in distributed heterogeneous systems resulting in dynamically and unpredictably changing risks. The SANCUS project aims to tackle these challenges through a ground-breaking design paradigm of a systematic and all-inclusive solution of true network protection. In particular, SANCUS proposes a solution based on a threefold firmware-runtime-optimisation analysis viewpoint, representing a three phase analysis.

Following we provide a detailed description of SANCUS's main contributions that can be summarized by the following steps: Step 1: (Model extraction) Dimension security and privacy as network attributes. This step is focused on the definition and modeling of security and privacy in terms of final formulas introducing the risks related to each device, application and development platform. Other attributes are taken into account: data throughput, energy efficiency, transmission latency, etc. Step 2: (Risk assessment) This step is focused on the definition of risk assessment of every network unit with the aim to derive mathematically precise protection recommendations.

The SANCUS approach also includes the implementation of the proposed model enriched with KPIs, performance and optimisation techniques including: the definition of new KPI metrics, which introduce the notion of cybersecurity and privacy as network performance measures; the design and development of automated security validation and verification solutions for extracting ground-truth risk evidence from both the code and network-level; and the introduction of optimisation mechanisms based on intelligent game theory decision-making frameworks and truly optimal defence recommendations subject to risks and vulnerabilities assessments in firmware, applications, network architectures and ICT deployment environments.

The paper is organised as follows. Section 2 presents the system architecture and connectivity mechanisms. Section 3 presents the pilots use cases illustrating the application of the proposed solution and Section 4 gives the conclusion and future work.

2 SANCUS SOLUTION ARCHITECTURE

2.1 Initial SANCUS System Architecture and Connectivity between the Engines

SANCUS has been initially approached as a new security suite to enable joint firmware and software validation and verification, as well as, security modelling and optimization by integrating six main security engines:

- (1) FiV – Firmware Inspection Validation engine,
- (2) CiV – Code Integrity Verification engine,
- (3) SiD – System Intelligent Defence engine,
- (4) MiU – Modelling of Individual Unit engine,
- (5) GiO – Game Implicit Optimization engine,

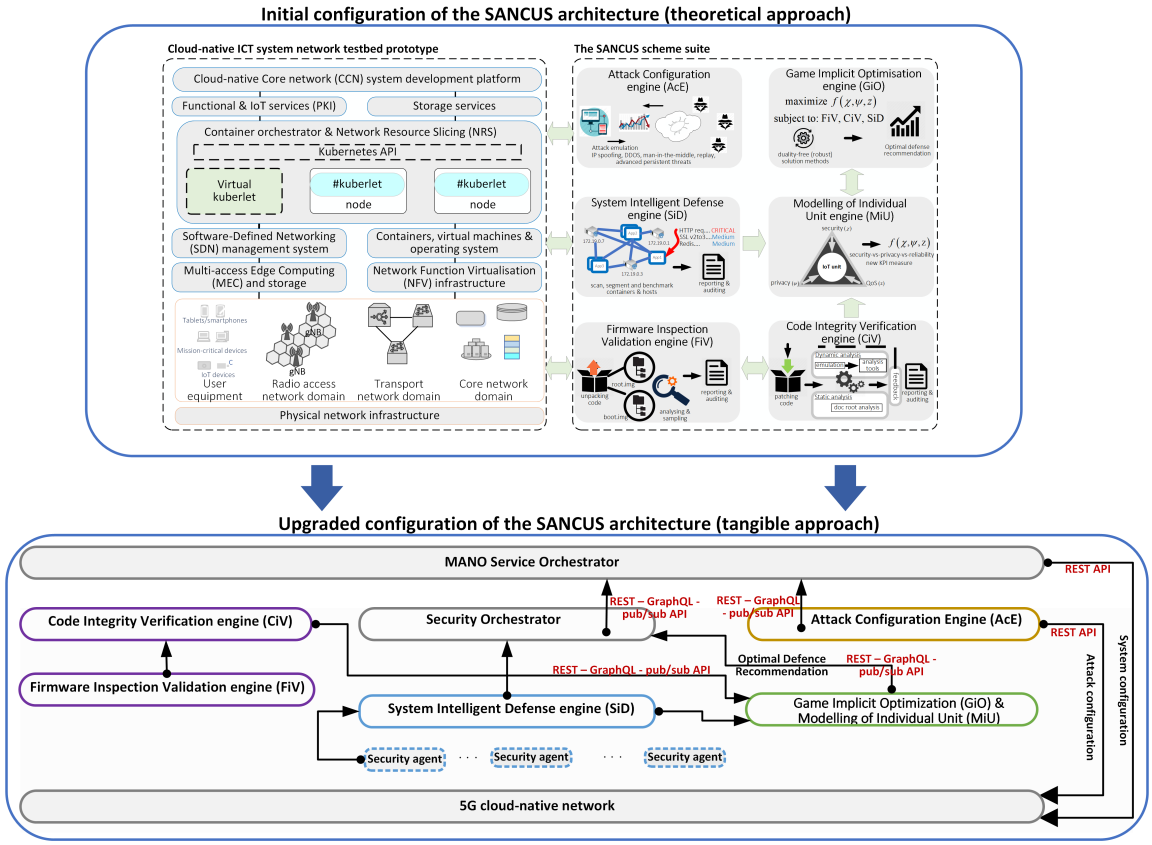

Fig. 1. Illustration of the SANCUS architecture. Top: initial approach for only extracting the optimal defence recommendation. Bottom: upgraded approach for extracting-and-applying the optimal defence recommendation.

- (6) AcE – Attack Configuration Engine.

In this respect, the architecture of SANCUS and the connectivity these engines had been initially approached theoretically, are illustrated at the top side of Fig.1. We note two important add-ons, such that the initial – theoretical – architecture can be integrated – tangibly – as a unified operational security suite. These add-ons are related to the addition of:

- (7) a Security Orchestrator, and
- (8) three Event Triggering Modules to be attached within (i) FiV/CiV, (ii) SiD, and (iii) MiU/GiO engines for automating the operation between these engines.

2.2 SANCUS engines and key functions

2.2.1 Firmware Inspection Validation engine (FiV). The proposed FiV engine will perform automated validation of firmware images at massive scale. FiV will combine different unpackers and samplers, allowing for searching through binary firmware images. After unpacking, it will attempt to recompose fragmented code portions for smoother and faster verification. The FiV operates in the following five phases:

(1) FiV classifies the firmware images in order to choose the appropriate tools for unpacking.
(2) The engine builds an extended pipeline of unpackers to gain deep inspection and starts unpacking.
(3) It sanitises potentially corrupted files to be ready for CiV analysis.
(4) It uses static analysis along with a set of heuristics to detect possible vulnerabilities.
(5) Any information found on vulnerabilities is sent to the appropriate engines, e.g. CiV, AcE.

2.2.2 Code Integrity Verification engine (CiV). The proposed CiV engine will perform automated verification and auditing of the unpacked OEM firmware images against risk factors. CiV will combine static (symbolic) and dynamic analysers to maximize the surface of vulnerability discovery, and optimise for scanning speed, testing precision, and validation efficiency. It will output information about these vulnerabilities to other engines, such as MiU, GiO, SiD, and AcE. The engine will go through the following steps:

(1) CiV analyses the firmware image provided by the FiV engine. It will select interesting targets based on heuristics, assisted by the FiV engine.
(2) The selected program is emulated in order to dynamically analyse it with fuzzing.
(3) When the fuzzer detects it is making little progress, static analysis by means of symbolic execution will be used to detect new inputs.
(4) After a certain amount of time has passed or when no new inputs can be found, the analysis of the program halts.
(5) The gathered information on vulnerabilities is sent to the AcE engine to confirm whether they can be exploited.
(6) If the AcE engine can confirm the exploitability of the vulnerability, a Common Vulnerability Scoring system (CVSS) like score is generated and sent along with the rest of the information to the MiU and GiO engines.

We expect that by combining static symbolic execution and fuzzing on an emulated target, the proposed CiV engine will be able to introduce a novel technique for firmware analysis beyond state-of-the-art solutions.

2.2.3 Software Risk Validation and Verification Engine (SiD). The proposed SiD engine is linked to the CiV and FiV engines for dimensioning different insights on vulnerabilities from both code and system levels. The engine will perform security validation for Docker open-source software development platform by providing new technique of automatically deploying lightweight distributed network analysis probes into each container to assess the robustness of the running software. In that way, SiD aims at continuous risk assessment including assets identification, vulnerabilities and threats detection, evaluating the risk, and providing list inventory of countermeasure and risk control options to be fused into the MiU modelling.

2.2.4 Modelling of IoT unit defining the security-vs-privacy-vs-reliability metric (MiU). The proposed MiU engine will step forward by analysing the outcomes of FiV, CiV and SiD to dimension the factors of cyber security, digital privacy and QoS reliability into the IoT unit and approach it as a three-node model determined within a multi-dimensional space of specific attributes. MiU relies on ground-breaking approach to express the trade-off between security-vs-privacy-vs-reliability by means of final formulas (explicitly), so as, the derived metric can be elaborated in any setting and case.

2.2.5 Game Implicit Optimisation Engine (GiO). The proposed GiO engine is directly linked to the MiU modelling for performing optimisation analysis to derive the optimal (final) SANCUS defence recommendation, i.e., maximise the performance of the security-vs-privacy-vs-reliability efficiency subject to the constraint that IoT devices, applications, end-users can coexist without any risk of being attacked and violating the applications requirements. In essence, GiO

stands on top from all the other SANCUS engines because it is to optimise the outcomes of MiU, which have been realised on the outcomes of FiV, CiV and SiD engines.

2.2.6 Attack Configuration Engine (AcE). The proposed AcE engine is directly related to the 5G test-bed prototype to deliver inclusive solution for modelling and emulating network container services and applications, along with network-wide attacks, forensic investigations, and tests that require a safe environment without the risk of proprietary data loss or adverse impact upon existing networks. AcE is planned to be designed as a highly customisable dynamic network modelling tool for enabling real-world virtualisation with real-time emulation of container network systems. The strength of this tool is that it will allow testing not only large-scale network infrastructures, but also emulating the end-users (IoT, routers, hotspots).

2.2.7 The SANCUS Security Orchestrator. The Security Orchestrator will be in charge of making a next-best action recommendation taking as input events triggered by the Service Monitoring and Analytics function, delivered as a result of monitoring and analysing changes in the status of the resources and devices. It will have:

- two inputs, i.e., from the SiD and the MiU/GiO engines related to the security agent placements (of SiD) and the optimal defence recommendation (of MiU/GiO), and
- one output to communicate with the service MANagement and Orchestration (MANO) system for coordinating the system resources towards implementing optimal system defence solution.

For this, the Security Orchestrator is necessary to integrate two main components, namely (i) a Security Policy Repository, which will store the security policies with reference to the service, slice and resource they apply to, and (ii) a Policy Engine. The Policy Engine will be in charge to performing the actual evaluation of policies by integrating (a) a rule engine that will be in charge to evaluating expressions in policy rules, (b) a policy information point that will be in charge to retrieving policy locators, policies and policy rules from the Policy Repository and evaluating them, resolving tags in expressions, (c) a context input module that will communicate with the monitoring and analytics modules of the MANO system and additional data and context sources, in order to retrieve relevant real-time information about the monitored resources, and (d) a context output module to communicate the optimal security recommendation performed by the MiU/GiO engines towards the entities in charge of actual enforcement upon the controlled resources (e.g. Virtual Infrastructure Management (VIM), Virtual Infrastructure Function Management (VNFM), etc.).

We see the SANCUS Security Orchestration as an automated process for deploying compute, storage, and networking elements. It will model the requests of the security engines into a manageable cloud-native infrastructure and will perform deployment and work plans according to the engines' requirements. The Security Orchestrator will interface to the service MANO components like, the VIM management and control, which offer access to the multi-site cloud environment. As such, the Security Orchestrator will fully abstract the infrastructure of the FiV/CiV, SiD, MiU/GiO engines, while its communication with the service MANO orchestrator will ensure engine isolation, i.e., the engines will not directly interact with the MANO infrastructure components, which limits access to the engines, thus limits attackers to compromise the engines. Furthermore, the Security Orchestrator will provide a "type system" to describe the means for creating or upgrading Security Policy Templates and offering possibility to define virtual instance templates. For example, characteristics of the security status of each networked component can be described within the Security Policy Templates properties, relationships, hardware and software requirements, policies, machine status, and deployment plans. The general idea is to define all the topics (e.g. protocols, equipment, environment variables, etc.) related with the network security and virtualization.

2.2.8 The SANCUS Event Triggering Module. After having described the upgraded SANCUS work-flow between the FiV/CiV, SiD, MiU/GiO engines and the newly added Security Orchestrator, we continue with the description of the Event Triggering Module, which will be attached to each engine. The Event Triggering Module will include (i) a manifest parsing sub-module, (ii) an event pre-triggering sub-module, and (iii) an event faking sub-module.

These sub-modules are essential mainly for the SiD engine due to its constant security monitoring and assessment, while in CiV/FiV and MiU/GiO not all sub-modules are important as the processes in these engines will be based on time-schedule (not constantly). To better understand the importance of the Event Triggering Module for SiD, we clarify the following. The manifest parsing sub-module is to parse the application manifest file for extracting API list of registered functions related to the application at hand. The Event pre-triggering sub-module is to send a request to the Event Faking sub-module for injecting corresponding false events when an application aims to use a system service, the callbacks of which will be registered through the API. The Event Faking sub-Module is to perform in-time event injection when received the request information for fake events. The fake events include not only callbacks listed in the application manifest, but also API callbacks that invoked at runtime. With all these sub-modules, the Event Triggering Module will be able to avoid being triggered by fake application events that can potentially be done on purpose by attackers to compromise the system. It will also notice the fake-related application to inform engines that this particular application may be the medium of an attacker to get deeper into the system. As such, the Event Triggering Module will be a reliable medium through which engines can communicate and triggered by each other on an event of an actual firmware bug and software vulnerability detection. Another key point is that besides avoiding fake triggering, the Event Triggering Module should be designed in a way to be compatible with a convenient API that is secured and familiar to developers. In this direction we will focus on designing the Event Triggering Module using either the REST or the GraphQL APIs. Next sub-Section describes the reasons why we will focus on choosing between these two APIs.

2.3 SANCUS main workflows

Having the aforementioned viewpoint in mind, let us now describe the upgraded work-flow by describing the functionality of each engine and the interaction between engines and Security Orchestrator with six Steps.

2.3.1 Step 1. At random time instance, the FiV engine will unpack a given set of firmware images by combining Binwalk [6], FRAK, Binary analysis [9] and radare2 [4] unpacking modules. After unpacking, FiV can perform pre-processing sanitisation, including firmware classification given that some unpacked firmware images may have been incompletely unpacked, or the location of the document root may not be obvious or corrupted.

2.3.2 Step 2. The unpacked software will be fused into the CiV engine that will perform static and dynamic analysis for detecting both firmware and application code vulnerabilities based on two effective methods, so-called, lazy computation and symbolic execution methods. The combination of those two methods with taint analysis and data-flow analysis, is marginally explored, yet it is likely to locate exploitable bugs in a rather reliable way compared to program slicing and type systems techniques as mentioned previously.

2.3.3 Step 3. In parallel to the operation of FiV and CiV, the SiD engine will perform continuous risk assessment by using a root cause analysis component to perform security validation of the software runtime deployment environment by means of (i) controlling the access to APIs, (ii) scanning control-plane and application-plane traffic for potential modifications and injections, and (iii) minimising the attack surface of each container for excluding code lines and

utilities that are not required by the application code, so as, each running container can be less useful to an attacker to compromise it.

2.3.4 Step 4. The outcomes of FiV, CiV and SiD will be managed by the MiU engine, so as, they will be expressed by means of mathematical optimisation criteria. Each criterion can be seen as a survey regarding the risk factors that influence security and privacy, hence, each criterion can be rated according to its degree of relative importance to another criterion on the basis of pair-wise comparison using properties of Analytic Hierarchy Process (AHP) concept and probability theory. In other words, given the fact that all detected risks have a degree of inherent uncertainty (that increases when more risks are related to each other) we will combine AHP and probabilistic weighting to implement the groupings of data risks related to each network device including those risk boundaries that are not sharply defined. On this basis, MiU will use utility theory for the conformal weighing of the AHP-related risks, and a technique, called nested radicals to dimension the weighted risks within an expected utility fitness function unified with the minimum throughput QoS requirements of each individual network device – recall from mathematics that the expected utility is the sum of the probability-weighted factors (like our AHP weights) of all possible final states of an option, where, in our case, the option's prospects are, in fact, integrated with a security-vs-privacy-vs-reliability asset level of each network device.

2.3.5 Step 5. As long as MiU will conclude the updated security modelling (e.g. based on the outputs of FiV/CiV and SiD), the GiO engine will be called to optimise security subject to these updated security modelling. The key challenge for GiO will be to capture the heterogeneity among all the created expected utility fitness functions of the network devices. It will do so, by integrating these functions as optimisation rules into Blotto and/or Stackelberg type competitive game problems, which will qualify the devices (or players) to autonomously compete for maximising their utilities using a common auction pool [2], thus, each device (or player) will state its own level of heterogeneity autonomously. The GiO game problem will also use Bayes-adaptive learning to supervise the reinforcement provided by the multi-variate probability distribution of the joint play, which is a formally elegant machine learning approach to realise optimal system behaviour in real time. The solutions to these complex, yet inclusive, problems will be derived by robust algorithms, which rely on intelligent duality-free solution techniques that we have shown to minimise the number of iterations for acquiring the optimal points [3].

2.3.6 Step 6. The output of GiO will be fused (via the context input module) into the Security Orchestrator, such that the latter can use its Policy Engine to create new Security Policy Templates, or upgrade existing templates, depending on the magnitude of changes required by GiO to protect the system. As long as the templates will be created/upgraded, the Security Orchestrator will use its context output module to provide these security templates to the service MANO Orchestrator – in essence, the templates will be instructions of how the MANO Orchestrator can move its Virtual Network Functions (VNFs) for protecting the system. Also, the Security Orchestrator will feature a constant input from SiD, such that it can be constantly informed for potential attacks and/or software vulnerabilities and act accordingly. Instead, the processes of MiU/GiO will take place in a non-constant – scheduled – basis, because even though the MiU modelling can be (theoretically) updated constantly, the GiO optimization process can only be (practically) resolved at scheduled time instances (e.g. every half hour or so) – in this meantime, the input of SiD can trigger alert for the Security Orchestrator to take pre-emptive measures for temporarily securing the system until GiO will announce the actual security defence recommendation.

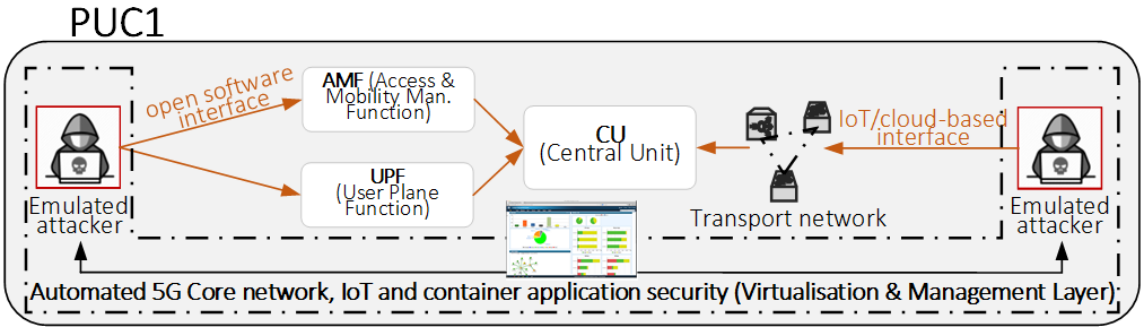

Fig. 2. PUC1- Automated 5G core network, IoT and container application and security.

3 SANCUS PILOTS

In this section, a list of use cases and attack scenarios is provided that will be used to demonstrate the effectiveness of the automated detection and protection tools developed in the SANCUS project against cyber threats. SANCUS identifies three Pilot Use Cases (PUCs) for catching up with its performance on Firmware Layer, Virtualisation Layer and Management Layer including their variations. The three PUCs are described following a structured way to uniformly elaborate on the description, requirements, actions and outcomes of each of the PUCs. For each of the PUCs a brief description, engaging modules (engines), pre- and post-conditions, and a basic flow or sequence of the actions are described. It should be noted that pre- and post-conditions refer to enablers of the PUCs and the result of the task(s) associated to each PUC, respectively.

3.1 Pilot Use Case 1

3.1.1 Brief Description. PUC1 focuses on representing the end-to-end services of 5G and IoT applications and network services. The goal is to perform a continuous risk assessment process in the network, end-user's equipment and virtualized infrastructure. PUC1's AcE performs as a network traffic and threat generator, modelling complex cyber-attacks in a service-based architecture and interfaces using the 5G virtualized core provided by NOKIA. Alongside, the SiD performs runtime monitoring over the system, in order to detect threats and vulnerabilities, and trigger the MiU and GiO engines, that allow PUC3 to perform mitigation actions. PUC1 aims to emulate and detect security incidents in the user equipment.

3.1.2 Modules. The involved platforms and engines in PUC1 are:

- Cloud-native Networked System: it comprises all the basic Network Function (NF) elements that are needed for common 5G applications. These elements are deployed as cloud instances of the relevant network functions through either open stack or containers virtualization approach. Figure 2 shows its Service-Based Architecture (SBA), where PUC1 will target all the components (user equipment, and transport and core networks) except for the radio access network. It is composed of the following modules: (i) 5G Equipment Identity Register (EIR), (ii) Network Repository Function (NRF), (iii) Network Slice Selection Function (NSSF), (iv)Authentication Server Function (AUSF), (v) Unified Data Management (UDM), (vi) Policy Control Function (PCF), (vii) Session Management Function (SMF), (viii) User plane function (UPF), (ix) Next Generation NodeB (gNB), (x) Access

and Mobility Management Function (AMF), (xi) Infra for supporting VMs and test automation, (xii) OpenStack Orchestrator, and (xiii) Containers Orchestrator.

- Attack Configuration Engine (AcE): it is used for configuring and emulating a cloud-native 5G network composed of a set of container services, applications and security mechanisms, as well as, acting as network traffic and threat generator that models complex cyber-attacks. It is composed of four modules, namely (i) system configurator, (ii) Traffic generator, (iii) Attack injector, and (iv) ML attack modeling and emulations.
- System Intelligent Defense (SiD) : The SiD engine provides automated validation and verification of the security of open-source systems and it dynamically assess the risks of running containers against threats, attacks, ransomware, viruses, breakouts and other suspicious activities. It is composed of the nine main modules to perform (i) Asset identification and classification, (ii) Vulnerability identification, (iii) Rule/ML-based threat identification, (iv) Root cause analysis, (v) Continuous risk value computation and forecasting, (vi) Alert reporting, (vii) Interfacing with the MiU/GiO engines, (viii) Monitor the network, using Security agents, and (ix) Presenting relevant visualizations of the collected and generated data, using a Security administrator dashboard

3.1.3 Pre-conditions. The first pre-condition for PUC1 is a cloud-native 5G network topology. The other pre-condition is pre-deployed security agents. As part of SiD, the security agents are distributed probes over the system, with the objective of measuring, collecting, analyzing and reporting events on the network. There are also two main inputs to the AcE engine including a collection of publicly available datasets of network vulnerabilities, and network topologies, and a collection of scripts to automate the deployment of attacks and topologies.

3.1.4 Sequence of actions. The sequence of actions for the System Intelligent Defence (SiD) engine is as follows.

(1) The security agents are first set through the security administrator dashboard (web-based application)
(2) The Asset identification and classification module identifies and classifies (i) the location, (ii) associated business processes, (iii) traffic at data elements, and (iv) threats and risks associated to each data element, in the network, end-users' equipment and the virtualized infrastructure
(3) Security agents monitor the network and send reports with network-based events to the Continuous risk value computation and forecasting module
(4) The Rule/ML-based threat identification module identifies existing and unknown threats, through the security agents deployed on the network. It includes (i) the identification of system attacks and security events, as well as (ii) the identification of current and future security issues
(5) The vulnerability identification module identifies threats and vulnerabilities on the system, relaying on the Common Weakness Enumeration (CWE) open database of vulnerabilities
(6) The Continuous risk value computation and forecasting module quantifies the risk of the detected threats, as a function of the likelihood, the magnitude of their impact, and the capability of applying mitigating controls at network level. Additionally, it predicts the impact of future risks in terms of joint security and cost
(7) Relevant dashboards are displayed to security operators
(8) The root cause analysis module performs a systematic analysis for identifying "root causes" of problems or events
(9) The alert reporting module produces reports of threats and vulnerabilities in raw format
(10) The MiU/GiO connector module formats the reports in matrix form and triggers the MiU/GiO engines in PUC3

The sequence of actions for the Attack Configuration Engine (AcE) is as follows.

(1) The System configurator deploys the targeted network topology, based on the collection of network topologies
(2) The traffic generator generates standard 5G network traffic
(3) The Attack injector generates malicious traffic, playing as an attacker based on the collection of network vulnerabilities and scripts to generate the attack message flow
(4) The ML attack modeling and emulations module combines the standard and malicious traffic and inject them into the network, emulating attacker/defender patterns

3.1.5 Post-conditions. The post-conditions for the PUC1 include (i) list of attacks on emulated traffic (AcE output), (ii) list of detected attacks (iii) root cause of detected attacks, (iv) list of alerts to be sent to the MiU and GiO, (v) forecast of detected risks impact in joint terms of security and cost, (vi) network status and visualization of risk level of the running ICT system on the dashboard.

3.2 Pilot Use Case 2

3.2.1 Brief Description. PUC2 focuses on performing the security compliance checks, as well as risk assessment on the IoT devices connected to SANCUS testbed on the firmware level. This PUC starts when firmware images from the end-user IoT devices are extracted. The firmware will be automatically analysed, and any actual security risks will be reported to the risk management platform based on the combination of outcome produced by the FiV and CiV engines. The extraction of the firmware is performed by FiV. During the extraction process sanitization techniques are used to prepare any potentially corrupted files of firmware to prepare them for further code integrity verification performed by CiV. The end result of the use case includes the risk report produced by a combination of FiV and CiV engines working together.

The goal of PUC2 is to examine the performance of FiV and CiV engines in how effectively and efficiently they can perform the firmware validation and verification process. No mitigation measures of any identified vulnerabilities will be applied within PUC2. The end result of the FiV and CiV validation process is the report containing risk assessment results. Moreover, the output of FiV and CiV will be used to verify whether MiU and GiO in combination can make decisions on risk acceptance / rejection or any other action to be implemented for identified risks. On top of that, the output of FiV and CiV might be provided to AcE as an input to perform validation of whether those vulnerabilities are exploitable on network level.

3.2.2 Modules. The following modules/engines are involved in PUC2: (i) FiV: Automated vulnerability inspection management engine, which combines static (symbolic) and dynamic analyzers to maximize the surface of vulnerability discovery, where multiple different unpackers and samplers will allow for searching through binary firmware images and recompose fragmented code portions for smoother and faster verification (ii) CiV: A new method for enabling precise and scalable taint analysis taken into consideration that taint analysis is a demand driven problem, which enables lazy computation of vulnerable information flows, instead of computing a complete data-flow solution, which is the reason for the traditional contrast between scalability and precision.

3.2.3 Pre-conditions. To perform PUC2 following pre-conditions should be satisfied: (i) The firmware has to be supported by the existing FiV's firmware unpackers and CiV's emulator (currently ARM64 with a potential expansion to MIPS) (ii) Location of engines: for PUC2 the server with FiV and CiV might be located on Secura's premises

3.2.4 Sequence of actions. The flow of the PUC is performed in following steps:

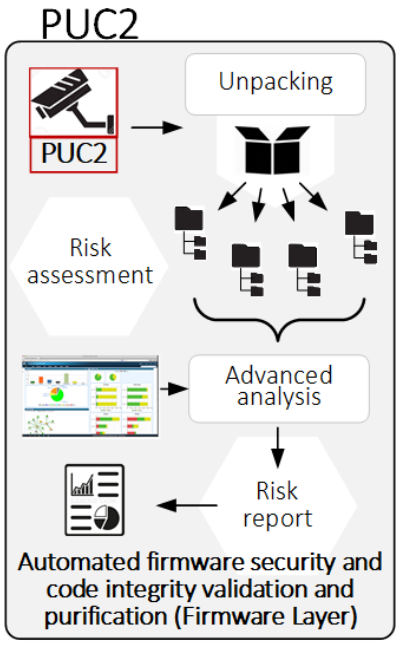

Fig. 3. PUC2- Security analysis of firmware.

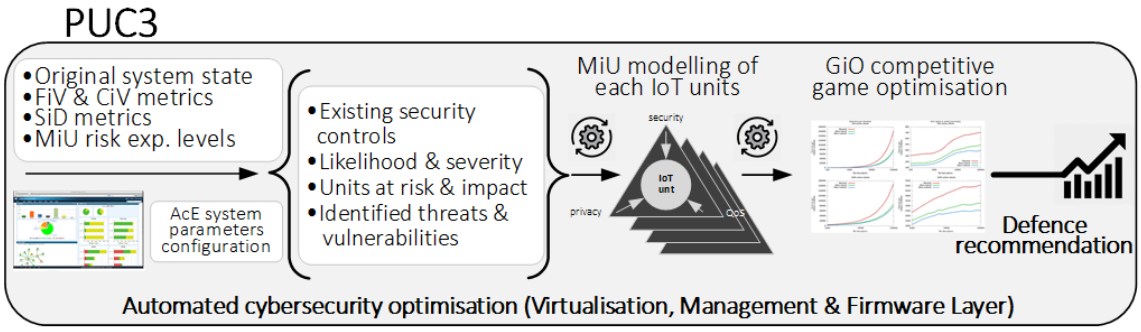

Fig. 4. PUC3- Automated cybersecurity optimisation (virtualisation, management and firmware layers).

- The firmware image is manually extracted from router
- The firmware image is uploaded on the server where FiV and CiV reside
- The firmware image is processed by FiV, the result is a thousand of unpacked files prepared for further analysis
- The extracted firmware files are automatically transferred to CiV after the unpacking process is over
- The extracted firmware files are processed by CiV, the results are a list of identified vulnerabilities with scores (in a form of the report)
- The report is manually transferred to MiU+GiO to perform PUC3

- Additional step includes manual transfer of the report to AcE as an input to try and exploit identified vulnerabilities.

3.2.5 Post-conditions. In the general case, a list of vulnerabilities will be provided to the 'user/operator' of the system. This data can be used by AcE and the MiU/GiO engines. The output includes: (i) List and description of vulnerabilities in the firmware (ii) Initial score for each vulnerability (without trying to exploit) based on CVSS (iii) A final score including results of exploitability for each vulnerability (a base is CVSS) based on the results of exploitability from AcE (iv) Information about vulnerabilities to form a decision on what needs to be done next based on a decision-making system.

3.3 Pilot Use Case 3

3.3.1 Brief Description. It is both time-consuming and costly to configure firewalls, routers, switches and other devices to minimise risks and to comply with security policies, especially for large networks. Effective security policy management also requires device-level analysis, network-level compliance analysis and risk assessment of ongoing changes, e.g. network, firmware. The design, deployment and support of such solutions for ensuring that the network can dynamically run optimally and that organisations can benefit from latest policy updates and solution offerings is essential. PUC3 will be used to validate the whole SANCUS scheme and its ability to support cybersecurity assessments at all examined layers (i.e. virtualisation, management, firmware) and evaluate its capacity for optimising the security-vs-privacy-vs-reliability performance of the network dynamically and in an automated manner.

3.3.2 Modules. The main modules involved in PUC3 are the MiU and GiO engines. The outcomes of FiV, CiV and SiD engines will be fused into the MiU engine, where they will be expressed in the form of mathematical optimization criteria. The methodology will rely on efforts in [10]. Each criterion includes the risk factors that impact on the security and privacy of network devices. Hence, each criterion can be weighted according to its degree of relative importance to another criterion. Subsequently, data risks related to each IoT device can be grouped. To this aim, the MiU may exploit some theories and techniques to weight the risks and to dimension the weighted risks within an expected utility fitness function unified with the minimum throughput QoS requirements of each end device.

The GiO engine, on the other hand, is responsible for optimizing the outcome of the MiU engine. The outcome of the MiU is forwarded for processing by the GiO engine, where the results will be optimized. The GiO engine captures the heterogeneity among all the created expected utility fitness functions of the IoT devices. This can be done by integration of these functions as optimization rules into some game problems, enabling the devices autonomously compete for maximizing their utilities.

3.3.3 Pre-conditions. The main pre-conditions (either directly or indirectly) to PUC3 are stemming from the FiV, CiV, SiD and AcE engines. After unpacking a given set of firmware images by FiV and performing pre-processing sanitisation, the unpacked firmware sources are fused into the CiV engine. The CiV engine later performs static and dynamic analysis for detecting both firmware and application code vulnerabilities where it is likely to locate exploitable bugs in a rather reliable way. CiV also creates full threat model on the 5G firmware and automated threat validation. Simultaneously, the SiD engine uses innovative continuous risk assessment mechanism to perform security validation of the docker runtime deployment environment. It then provides list inventory of countermeasures and risk control options to be fused into the MiU engine. In other words, the outcomes of FiV, CiV and SiD are processed by MiU engine. Finally, the AcE engine will emulate the defined PUCs by introducing a set of network modelling and emulation module,

traffic and attack generators, abstraction module for IoT devices, and Big Data analytics and visualization for automated labelling and processing of huge amount of gathered data. Along with the other engines, the original system state in terms of the value of the different state variables is also a pre-condition to PUC3. The above populates existing security controls, likelihood and severity of vulnerabilities, units at risk and impact, identified threats and vulnerabilities.

3.3.4 Sequence of actions. As depicted in Figure 4, pre-conditions populate data in PUC3 considering the following information flow. Initially, the original state of the system is identified in terms of the value of different state variables, for instance, connected platforms, number of devices and end-users. Then the SiD engine triggers the software continuous assessment process to obtain the cyber-risk metrics, e.g. successful attack rate, software patches, new or modified scripts. Moreover, the cyber-risk metrics (for instance, unauthenticated devices, outdated network components) from the FiV/CiV and SiD engines are obtained. Having received this information, the MIU and GiO module trigger.

Considering the input data and the configuration, the MiU engine models the system to identify the risk exposure levels of the system. This includes the quantification of security parameters of network devices with respect to the communication protocols, data traffic, the type of encryption used for their applications and users' data, etc. The MiU makes estimation of the likelihood of the cyber-attacks associated to each device in the network. Then the GiO engine makes optimal recommendations regarding the cyber-security control. The recommendation is quantified by the security-vs-privacy-vs-reliability maximisation of the GiO engine. It should be noted that various cyber-attack scenarios, network traffic loads, topologies and respective cybersecurity optimisation cases will be considered and examined. Finally, the MiU and GiO modules, forward the output to the security orchestration module in order to make a decision on where the security agents should be placed.

3.3.5 Post-conditions. Since PUC3 has two modules, in this section we briefly explain the post-conditions of both modules and the PUC3. The MiU aims at expressing its feature modelling by means of final formulas, and shapes expected utility objectives for arranging the fitness functions of all IoT units. This function represents the probability-weighted fitness functions of all possible final states of the security-vs-privacy-vs-reliability performance of the network overall. The post-condition for the GiO engine is designing the optimal defence recommendation. This is done, by maximizing the performance of the security-vs-privacy-vs-reliability efficiency subject to the probability that IoT devices, applications, end-users can co-exist without any risk of being attacked. All in all, there will be a validation for the SANCUS decision-making scheme for its potential to recommend realistic, affordable baseline measures that maximize security, privacy and QoS reliability jointly, along with improving the ICT system resilience and sustainability.

3.4 Example Threat Scenarios

We have considered three threat levels characterizing the severity of the threats. Once FiV/CiV and/or SiD/AcE engines detect a vulnerability within the firmware or code or traffic, a ranking will be made and fed to the PUC3 engines for decision making. These outputs include list and description of vulnerabilities in firmware, or attacks in emulated traffic, initial and final score for each vulnerability, list of detected attacks and their root cause, list of alerts and forecast of the detected risks impacts on cost. In the following, we list potential threat scenarios that may be considered under PUC3:

3.4.1 Low threat severity: In PUC3, a threat scenario of low severity is considered when the severity of the threats in FiV/CiV and/or SiD/AcE engines' managed threats are also of low level, e.g. exploitation of the freed memory threat, null pointer dereference threat, DoS, eavesdropping and abuse of authentication. Specifically, a combination of engines is considered in order to understand how a change in the firmware of a device and/or the software will be detected by

the FiV/CiV and SiD engines, respectively. In the assumption that a null-pointer dereference vulnerability is detected by the FiV/CiV, the MiU/GiO may recommend to perform sanity checks on pointers. Similarly, if SiD detects a DoS attack the MiU/GiO may recommend the instantiation of a security control (e.g. firewall) in the infrastructure. Depending on the nature of threats the MiU/GiO will recommend an optimal defence countermeasure for manual application.

3.4.2 Medium threat severity: A threat scenario of medium severity is considered when the severity of the threats in the SiD/AcE engines are of medium level. However, the MiU/GiO engines' recommendations are forwarded to the security orchestrator for enforcement of the recommendations by the security and MANO orchestrator. For instance, in the case of a man-in-the-middle attack detected by the SiD/AcE engines the MiU/GiO will propose an optimal strategy for protecting the privacy of data. The recommended strategy will be forwarded to the security and MANO orchestrator for the automated enforcement of the strategy.

3.4.3 High threat severity: A threat scenario of high severity may be considered when the MiU/GiO engines will receive inputs from multiple engines and for an optimal recommendation to be taken and enforced in a (semi-)automated manner by the security and MANO orchestrator. The scenarios in this case may have to cope with high severity threats detected by the individual engines. Similarly to the low severity scenarios, the MiU/GiO engines will recommend an optimal decision ensuring the security, privacy and availability of the SANCUS architecture. The optimal recommendations will be enforced in an automated manner, when the recommendations can be applied on the virtualised infrastructure via the instantiation of security controls and/or re-organisation of existing virtualised services. Manual intervention may be required for the enforcement of recommendations that may not be supported by the security and MANO orchestrator, e.g. firmware patching.

4 CONCLUSION

In this paper we have presented the main aspects of the SANCUS project. SANCUS proposes an innovative solution based on the combination of different techniques in order to obtain secure, trustworthy and reliable 5G networks. It provides a tooled framework that brings together different engines enabling next-generation 5G systems to analyse and validate their firmware, network applications and services. SANCUS also proposes risk assessment techniques in order to improve network application and service security, privacy and reliability. This paper further presents an overview of the SANCUS architecture, including a theoretical and a realistic version, and the pilots use cases to which the proposed solution will be applied.

ACKNOWLEDGMENTS

This research is supported by the H2020 SANCUS project under the agreement number 952672.

REFERENCES

[1] Ijaz Ahmad, Tanesh Kumar, Madhusanka Liyanage, Jude Okwuibe, Mika Ylianttila, and Andrei Gurtov. 2018. Overview of 5G Security Challenges and Solutions. *IEEE Communications Standards Magazine* 2, 1 (2018), 36–43. https://doi.org/10.1109/MCOMSTD.2018.1700063

[2] Su Fong Chien, Charilaos C Zarakovitis, Qiang Ni, and Pei Xiao. 2019. Stochastic asymmetric blotto game approach for wireless resource allocation strategies. *IEEE Transactions on Wireless Communications* 18, 12 (2019), 5511–5528.

[3] Su Fong Chien, Charilaos C Zarakovitis, Ken Roberts, and Tiew On Ting. 2017. Sharp approximation for transcendental equation in optimising energy efficiency. In *2017 IEEE 13th Malaysia International Conference on Communications (MICC)*. IEEE, 90–95.

[4] Ang Cui. 2012. Embedded Device Firmware Vulnerability Hunting with FRAK. *DEF CON* 20 (2012), 9.

[5] Mohamed Amine Ferrag, Leandros Maglaras, Antonios Argyriou, Dimitrios Kosmanos, and Helge Janicke. 2018. Security for 4G and 5G cellular networks: A survey of existing authentication and privacy-preserving schemes. *Journal of Network and Computer Applications* 101 (2018), 55–82. https://doi.org/10.1016/j.jnca.2017.10.017

[6] Craig Heffner. [n.d.]. *Binwalk – firmware analysis tool designed to assist in the analysis, extraction, and reverse engineering of firmware images.* https://tools.kali.org/forensics/binwalk

[7] Rabia Khan, Pardeep Kumar, Dushantha Nalin K. Jayakody, and Madhusanka Liyanage. 2020. A Survey on Security and Privacy of 5G Technologies: Potential Solutions, Recent Advancements, and Future Directions. *IEEE Communications Surveys & Tutorials* 22, 1 (2020), 196–248. https://doi.org/10.1109/comst.2019.2933899

[8] Rajaneesh Sudhakar Shetty. 2021. *5G Packet Core Testing Strategies.* Apress, Berkeley, CA, 235–275. https://doi.org/10.1007/978-1-4842-6473-7_5

[9] Tjaldur Software Governance Solutions. [n.d.]. *Binary Analysys Toolikit.* http://www.binaryanalysis.org/

[10] Charilaos C. Zarakovitis and Qiang Ni. 2016. Maximizing Energy Efficiency in Multiuser Multicarrier Broadband Wireless Systems: Convex Relaxation and Global Optimization Techniques. *IEEE Transactions on Vehicular Technology* 65, 7 (2016), 5275–5286. https://doi.org/10.1109/TVT.2015.2455536