PROJECT FINAL REPORT

Grant Agreement Number:	609734
Project Acronym:	ASPIRE
Project Title:	Advanced Software Protection: Integration, Research and Exploitation
Funding scheme:	Collaborative project

Name, title and organisation of the scientific representative of the project's coordinator:

Prof. Dr. Bjorn De Sutter
Ghent University
Technologiepark-Zwijnaarde 15
B9052 Gent

Tel:	+32 9 264 33 67
Fax:	+32 9 264 3594
E-mail:	coordinator@aspire-fp7.eu
Project website address:	https://www.aspire-fp7.eu

Editor

Bjorn De Sutter (UGent)

Contributors (ordered according to beneficiary numbers)

Bart Coppens, Vicky Wandels (UGent)

Cataldo Basile (POLITO)

Brecht Wyseur (NAGRA)

Mariano Ceccato (FBK)

Paolo Falcarin (UEL)

Michael Zunke (SFNT)

Jerome D'Annoville (GTO)

The ASPIRE Consortium consists of:

Ghent University (UGent)	Coordinator & Beneficiary	Belgium
Politecnico Di Torino (POLITO)	Beneficiary	Italy
Nagravision SA (NAGRA) (Third Party: EDSI)	Beneficiary	Switzerland (France)
Fondazione Bruno Kessler (FBK)	Beneficiary	Italy
University of East London (UEL)	Beneficiary	UK
SFNT Germany GmbH (SFNT)	Beneficiary	Germany
Gemalto SA (GTO)	Beneficiary	France

Coordinating person:	Prof. Bjorn De Sutter
E-mail:	coordinator@aspire-fp7.eu
Tel:	+32 9 264 3367
Fax:	+32 9 264 3594
Project website:	www.aspire-fp7.eu

Disclaimer
The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 609734.

Contents

Section 1 Final Publishable Summary Report

1.1 Executive Summary

Project name:	**ASPIRE**	*Start date:*	1st November 2013
Grant Agreement:	**609734**	*Duration:*	36 months
Project Website:	www.aspire-fp7.eu		
Contact:	coordinator@aspire-fp7.eu		

The mission of ASPIRE *was to integrate state-of-the-art software protections into an application reference architecture and into an easy-to-use compiler framework that automatically provides measurable software-based protection of the valuable assets in the persistently or occasionally connected client applications of mobile service, software, and content providers.*

Motivation For mobile devices like smartphones and tablets, security solutions based on custom hardware (as is traditionally done with, e.g., smart cards, set-top boxes, and dongles) are not feasible anymore. Software protection is therefore utterly important; it can be a maker and a breaker. Current software protection techniques are incredibly hard to deploy, cost too much and limit innovation. Stakeholders in mobile devices need more trustworthy, cheaper software security solutions and more value for the money they spend on software security. In this project, three market leaders in security ICT solutions and four academic institutions joined forces to protect the assets of one class of stakeholders: the service, software, and content providers. From their perspective, mobile devices and their users, which can engage in attacks on the software and credentials installed to access the services or content, are not trustworthy.

Final results and their potential impact and use The software protection technology that has been developed consists of (i) the ASPIRE reference architecture for combining and composing multiple layers and types of software protections; (ii) designs and implementations of a range of online and offline protections, some of which pre-existed, some which are new or significant improvements over the previous state of the art; (iii) the robust ASPIRE Compiler Tool Chain that enables the automated, combined deployment of selected protections on real-world use cases; (iv) the ASPIRE Decision Support System and its ASPIRE Knowledge Base to assist the user of the tool chain with the selection of the protections best suited to protect the user's software and the assets embedded in it; (v) the ASPIRE software protection evaluation methodology to assess the value of software protections vis-à-vis man at the end attacks. A large part of the developed software prototypes is available as open source with extensive documentation, and more than 20 demonstration videos have been published. A significant part of the research has already been peer reviewed, many additional papers are still in the pipeline. Through keynotes and tutorials, incl. in workshops organized by the Consortium, the European software protection community has been revitalized and has been made well aware of the project and its results.

Some of the project results are already ready for **commercial exploitation**. A spin-off is in the making at FBK, and a technology transfer from UGent to industry has already taken place. Some of the specific protections developed within the project are used in products in the pipeline in business units of the industrial partners. As such, the project **strengthens the position of European companies**, incl. of course the project partners, whose business models depend on securing the assets embedded in their software. Other results are not ready for immediate commercialization. But with the whole ASPIRE Framework encompassing the compiler tool chain, the decision support system, many protections, and tools that automate the application of the software protection evaluation methodology, **the consortium has demonstrated that measureable, assisted deployment of software protection is feasible**. The open source availability of the Framework will **help the European R&D community** to bridge the gap to commercial deployment of the ASPIRE approach, not in the least by providing all the foundational infrastructure necessary for complementing and expanding the expert knowledge that was already gathered in the project from the PIs' expertise, from professional penetration tests, from the pubic challenge, and from external advice.

1.2 Project Context and Objectives

1.2.1 Context

ASPIRE addressed major issues concerning the security of mobile services in application domains like multi-screen mobile TV, software licensing, and credentials and sensitive data stored on mobile devices:

- For mobile devices like smartphones and tablets, security solutions based on custom hardware (as is traditionally done in these domains with, e.g., smart cards, set-top boxes, and dongles) are not feasible anymore.
- Software protection is therefore utterly important; it can be a maker and a breaker.
- Current software protection techniques are incredibly hard to deploy, cost too much and limit innovation.
- Stakeholders in mobile devices need more trustworthy, cheaper software security solutions and more value for the money they spend on software security.

In this project, three market leaders in security ICT solutions and four academic institutions joined forces to protect the assets of one class of stakeholders in mobile services: the service, software, and content providers. From their perspective, mobile devices and their users, which can engage in attacks on the software and credentials installed to access the services or content, are not trustworthy.

1.2.2 Strategic Objectives

With the ASPIRE solutions, we wanted mobile software security to become

- **Trustworthy** by leveraging on the available network connection and developing a layered security approach of strong protections;
- **Measurable** by developing practical, validated attack and protection models and practical metrics;
- **Cheaper** by integrating support for the protections into the productizable ASPIRE security framework;
- **More valuable** by enabling shorter time-to-markets and technology that is applicable to more cases.

Our ultimate goal was to provide software protection that is equally strong as existing hardware-based protections, such as custom set-top boxes, smart cards, and dongles. We therefore proposed to develop strong protections techniques and metrics along five mutually strengthening lines of defence, and to integrate support for them into the ASPIRE Tool Chain and Decision Support System, which we would demonstrate and validate on three real-world use cases from the industrial partners in the mentioned domains, and in a public challenge.

1.2.3 Overall Strategy

The overall strategy of the work plan is depicted in Figure 1.

WP1 defined the application requirements to drive the project's research and technology development (RTD), its validation, and its demonstration on three industrial use cases. The security requirements were unified into an attack model and a reference architecture of protected applications. The attack model, the architecture, and the requirements then were used to validate the application of the ASPIRE tool chain on the use cases.

The core ASPIRE RTD was situated in **WP2**, **WP3**, **WP4**, and **WP5**.

- In **WP2** and **WP3**, new, concrete techniques for software protection were designed and developed for each of the five identified lines of defence: data hiding, algorithm hiding, anti-tampering, remote attestation and code renewability. **WP2** focused on techniques that are applicable offline. **WP3** focused on online techniques in which the application is split over a client-side component and a server-side component running on a secure, trusted server. Gradually, tool support for the techniques was integrated into the ASPIRE framework.

- In **WP4**, a security evaluation methodology was developed for the software protections developed in WP2 and WP3. This methodology covers the attack model developed in WP1, and concentrates on the link between applied protections and the delay they cause for the attacker. To make the evaluation measurable, a software metric approach was designed and developed. Based on practical insights and experiments with attack tools and human attackers, a knowledge base was set up and tuned.
- **WP5** brought together and integrated all developed technology to ensure their compatibility and composability. The tools that apply the protections developed in WP2 and WP3 were integrated in the ASPIRE Compiler Tool Chain. The developed security evaluation technology and knowledge base of WP4 was integrated into the ASPIRE Decision Support System.

In **WP6**, the three uses cases defined in WP1 were implemented and protected by means of the ASPIRE framework to evaluate the framework, to demonstrate its capabilities, and to ensure a clear path to exploitation.

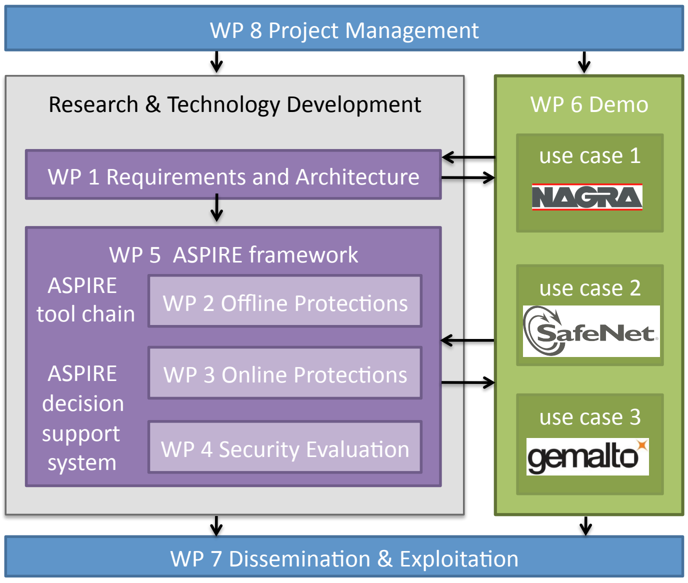

Figure 1 - Overall project strategy

1.2.4 S&T Objectives

1.2.4.1 WP1 Requirements and Architecture

In year one, WP1 had two objectives. The first one was to **specify three industrial use cases** and their attack models. One use case per industrial partner was foreseen, as follows:

- **Task T1.1: Secure integration of DRM libraries** (NAGRA)
- **Task T1.2: Any end-point software licensing** (SFNT)
- **Task T1.3: Software-based security for credentials** (GTO)

The second early goal, in Task T1.4, was to derive and define a generic **attack model**, the **application-level security requirements**, and a common **protection reference architecture** that would drive the other WPs, all based on the use case designs and their attack.

Towards the end of year 2, the objective of Task T1.5 was to **validate** whether or not the project solutions (reference architecture, protection techniques, tool chain, decision support system, security evaluation methodologies) as developed thus far in the other WPs were able to protect the use cases within the attack model and the security requirements. Moreover, the attack model, the security requirements description and the reference architecture were to be **updated** to reflect lessons learned and to ensure they were still up to date.

At the end of the project, the **final validation** had to cover all solutions developed in the other WPs, and incorporate lessons learned and insights obtained from all the experiments with academic hackers, with professional pen testers, and in the public challenge.

1.2.4.2 WP2 Offline Software Protections

The goal of this WP was to develop offline protections. For some, the goal was to push the state of the art, for others the goal was to have basic implementations that could be integrated with the other protections in WP5 to study composability. The WP consisted of five tasks, with the following objectives.

Task T2.1: Data obfuscation: Develop stronger data obfuscation techniques by making them depend on dynamic properties such as aliasing or hard to analyse features. Develop a source-level tool set to support the automated application of these techniques and of all other source-level techniques of this project.

Task T2.2: White-box cryptography (WBC): Design and implement provably strong WBC solutions based on the decomposition of multivariate equations over finite fields as an extension to a provided WBC library generator tool. Extend that existing WBC tool and library to support dynamic keys. Study the potential for using weaker but faster WBC schemes in time-limited protections, where renewability can ensure that keys are replaced before being broken. Use WBC to protect the bytecode used in the virtual machine (VM) we will embed in client-side applications. Use WBC in an indirect way in a diversified cryptographic library by protecting application keys by a key that is derived from a master key protected by WBC.

Task T2.3: Client-side code splitting: Develop a code splitting approach that can extract native client application code to replace it by bytecode that is interpreted in an embedded VM to protect against code inspection.

Task T2.4: Binary Code Obfuscation: Develop support for existing control flow obfuscation for the project's use cases and platforms. Extend the application of control flow obfuscations to cover both the original application code and the code of the embedded VM and other security components, such that the total binary is not easily decomposed into its two components by an attacker. Develop domain-specific obfuscations to protect cryptographic algorithms against pattern-based static or dynamic attacks.

Task T2.5: Anti-tampering: Develop anti-debugging techniques in which the client-side application is partially transferred into a debugger that gets attached to it, such that that debugger cannot be removed to make room for a malicious one. Develop limited overhead call stack checks and control flow checks to detect inappropriate callbacks from external libraries not protected by the other techniques. Develop code guards of which the functionality is split over the binary code and the bytecode, and that also guards both the application code and the embedded VM's code.

1.2.4.3 WP3 Online Software Protections

The goal of this WP was to design and develop state-of-the-art online protections. The WP consisted of three tasks, with the following objectives:

Task T3.1: Client-Server Code and Data Splitting: Develop a code splitting approach that can extract part of the client application code to execute it on a trusted server where it cannot be observed by an attacker. Combine server-side execution with execution in a client-side VM by having a server deliver the so-called mobile code to the client. Research data splitting techniques for protecting larger data sets by splitting the data over server and client.

Task T3.2: Remote attestation: Build on code splitting to develop a new type of implicit attestators (that relies on seemingly normal data being transferred from client to server and on the partial execution on the server) to make authenticity verdicts. Study advanced remote attestation aspects to improve the applicability, scalability, and dependencies of implicit attestation. Develop so-called time bombs to force the clients to connect to the server to undergo attestation regularly. Develop specific attestation schemes that can detect code cloning.

Task T3.3: Renewability: Extend the server-side functionality of the ASPIRE reference architecture to support renewability in time, i.e., to make the mobile protection code that is provisioned by the server to the client vary over time. Diversify the bytecode representation in the VM in space, to ensure that an attack on one application copy cannot be repeated immediately on another copy. Extend the existing binary code diversification contributed by UGent through Diablo from a purely stochastic process to a controlled process. Use a controlled process and optimization approaches to generate maximally diversified application copies.

1.2.4.4 WP4 Security Evaluation

The overall objectives of WP4 were the development of a software protection evaluation methodology and the necessary models and metrics to make software protection measurable. Penetration tests of various forms will be conducted to populate the models and better understand the relation between attacks and protections. The goals of the five tasks were as follows.

Task T4.1 Security Model and Evaluation: Develop a formal security model, security evaluation methodology, and knowledge base in which the relation between protections and attacks is incorporated such that the model can predict the delay an attacker will incur on an attack given a description of the ASPIRE protections applied and the assets of the program to be protected. Incorporate all information obtained from the experiments in T4.2–4 in the model.

Task T4.2 Complexity metrics: For a range of attacks, analyse modes of operation in attacks to define the heuristics used by attackers to steer their activities during attacks. Develop scripts that implement those heuristics and automate those activities that involve (semi-)automated attacker tools. Design complexity metrics based on the data set sizes observed during and after the execution of those scripts. Analyse how protection techniques being applied increase those data set sizes to estimate the delay an attacker will incur as the result of a protection.

Task T4.3 Experiments with academic subjects: Iteratively design, execute, and analyse controlled experiments in which PhD and MSc students attack protected programs to evaluate the delay resulting from the protections.

Task T4.4 Experiments with industrial tiger teams: Similar to Task T4.3, but in this task, industrial experts in tiger teams perform the attacks.

Task T4.5: Public Challenge: Setup a public challenge, run and monitor the challenge, analyse the (reported) attacks and their failure or success.

1.2.4.5 WP5 Framework

The overall objectives of WP5 were to develop a tool chain to deploy all the developed protections, a decision support system to assist the user of the tool chain, and open source versions thereof.

Task T5.1: ASPIRE Compiler Tool Chain: Design the ASPIRE tool chain: the set of tools that will be used as plug-ins, all APIs, information formats and semantics, the overall flow of the tool chain, etc. Ensure that it runs on the use case applications. Integrate all technology developed in WP2 and WP3 into this tool chain. Develop an easy-to-use GUI interface.

Task T5.2 ASPIRE decision support system: Develop a tool that allows a (non-expert) tool chain user to invoke the tool chain and obtain a selected level of protection. Develop a tool module that interfaces the model and knowledge base of Task T4.1 to obtain a list of tool chain configurations (combinations of protections to apply and their parameters) that will meet the user's requirements. Develop an optimization module that finds the optimal tool chain configuration for a target optimization function. Develop a GUI and integration with the tool chain.

Task T5.3 Open Source: Isolate, separate and clean up those parts of the ASPIRE framework that can be open-sourced. Open-source them with a manual.

1.2.4.6 WP6 Use Cases

To objectives in this WP were to develop and prepare the three use cases that are to be used to validate, test, and tune the ASPIRE framework and its components, and to demonstrate the ASPIRE results.

1.3 Main S&T Results / Foregrounds

1.3.1 WP1 Requirements and Architecture (M01-M36)

In Tasks T1.1–3, the industrial partners defined **three use cases** that are representative for their business applications and that embed similar assets to be protected. The use cases themselves were implemented in WP6, and the elicited security requirements were bundled and extended in Task T1.4. Importantly, the project opted for **more complex** use cases, i.e., **dynamically linked native Android ARMv7 libraries** loaded into processes largely out of control of the project, to ensure their representativeness of real-world applications, rather than going for simpler implementations that would have minimized the engineering effort needed in the other WPs.

1.3.1.1 Task T1.4 Attack model, security requirements, reference architecture

The ASPIRE Attack Model (D1.02, 89 pages, confidential)

The model first describes the scope of attacks on software assets against which the ASPIRE project developed protections. The **attack scope** is limited to **MATE attacks on native binary code** that is normally executed on mobile devices. Attackers can, however, also attack the software in their labs, where they have a whole range of tools, techniques, and more accessible devices at hand. With their toolbox, they can perform white-box attacks on the software and its assets. To put the attacks and the attackers' goals in perspective, the project builds on an **economic attack model for MATE attacks**, in which attackers try to gain more from exploiting an attack than they need to invest to engineer it. **Obviously, the ASPIRE protections then aim for requiring more effort from the attacker, and for lowering his potential gains.**

The attack model also describes t**he assets** that attackers might be interested in and that the ASPIRE technology aims to protect. These assets include **private data, public data, unique data, global data, traceable code and data, code, and code execution**. Their information security attributes that need to be protected are **confidentiality, privacy, integrity, non-repudiation, and execution correctness**. An overview is also presented of the threats that attackers can pose on these assets and attributes.

A large section of D1.02 is devoted to extensive, exhaustive discussions of the known attacks and attack steps that attackers deploy to attack the previously discussed assets. The list of attacks was assembled following a consortium-wide literature study covering academic and industrial experience within the project partners, academic literature, and online forums and hacker blogs. **Static attacks** are considered, in which no software under attack is executed, as well as **dynamic attacks**, that involve execution the software on selected inputs, and **hybrid attacks**, that combine static attack features with dynamic attack features. The static attacks are classified into **static structural code and data recovery**, **structural matching of binaries**, and **static tampering attacks**. The model also discusses the most important **attack attributes**: identification and exploitation. **Identification** refers to the expertise needed by attackers to engineer an attack. **Exploitation** refers to the damage that can be done on a provider's or vendor's business once a successful attack is engineered. The dynamic attacks are classified into **attacks on communication channels**, **fuzzing**, **debugging**, **dynamic structure and data analysis**, and **dynamic tampering attacks**. For all classes of attacks, the state of the art, the available tools and mitigations, and the attributes are discussed.

Finally, **six concrete attack** paths are presented to clarify how the different types of attacks are in practice combined to attack assets in real world use cases.

All partners consider the ASPIRE Attack Model a very useful S&T result, that allows them to put their S&T results of the other WPs in the correct perspective. The model remains confidential because industrial partners do not want such a valuable overview of available attack methods to fall in the hands of potential attackers of their products.

The ASPIRE Security Requirements (D1.03, 18 pages, confidential)

The document lists 42 security requirements and recommendations for the ASPIRE tool chain and for protected applications. They reflect requirements **real-world applications** (ideally) meet. Three categories of security requirements are elicited: functional, non-functional, and assurance requirements. **Eleven functional security requirements and recommendations** elicit which security functionalities an ASPIRE protected application needs to contain. For example, they need to comprise some authentication functionalities or remote attestation techniques need to be used. **Thirteen non-functional security requirements and recommendations** express the properties the ASPIRE projected application need to contain. This relates to inherent security properties such as obfuscation techniques that need to be present, or the fact that no debug information must be in the system. **Eighteen assurance security requirements and recommendations** relate to the ASPIRE tool chain itself and the life cycle of the protected application. For example, the fact that the ASPIRE tool chain needs to contain versioning support, accept parameters that allow further tuning of a trade-off between performance and security, and must have appropriate logging features.

The list of ASPIRE security requirements and recommendations is an important S&T result. It formed one of the foundations of the successful validation towards the end of the project.

The ASPIRE Reference Architecture (D1.04, 99 pages, public)

The ASPIRE Reference Architecture defines the components of ASPIRE-protected client-side applications and their server-side support, and how these interact with each other. In other words, this reference architecture presents which components are introduced by ASPIRE protection techniques and how these operate during the execution of the protected application.

As a basis, a multi-tier architecture is defined and documented. This captures an architecture where a multitude of client applications can connect to the ASPIRE portal infrastructure, which will manage the connections with a multitude of backend services. Within the context of this multi-tier architecture, the ASPIRE protection techniques are defined and their architecture is detailed. This includes the server-side and client-side components that each technique introduces, and how those components operate and communicate during the execution of the protected application. The document also presents the ASPIRE protocol, the ASPIRE server portal, and the ASPIRE Client Communication Logic through which all online ASPIRE protected applications communicate with the ASPIRE security server. Two different types of protocols are included: a simple request protocol and a protocol based on WebSockets.

Next, the document presents the architecture of the ASPIRE anti-reverse engineering protection techniques. These include obfuscation techniques and anti-debugging techniques. With respect to the architecture of the ASPIRE anti-tampering techniques, the architecture does not present complete solutions as individual techniques in separate subsections. Instead, different types of components are individually presented: tamper detection components (attestator components and verifier components) and tamper response components (delay components and reaction components). A complete anti-tampering solution comprises these different types of components. As tamper detection technique, code guards, CFG tagging, call stacks check, and anti-cloning are presented. As response components, delay data structures and software time bombs are presented. Some examples of compositions thereof are introduced as well: a completely offline combination, and remote attestation techniques.

The reference architecture also contains an assessment of the composability of the many protections supported by the ASPIRE Compiler Tool Chain, i.e., to what extent multiple protections can be applied to protect the same code fragment. It also discusses where synergies exist between individual protections to let them reinforce each other, and where additional design and development work was invested in building even stronger protections out of compositions of existing ones. This specifically concerns adaptations to mobile code, remote attestation (and its code guards) and client-side code splitting to support remote attestation of mobile code & data.

Finally, the reference architecture details the forms of renewability that were developed.

A paper on the ASPIRE reference architecture has been published in the 13th Working IEEE/IFIP Conference on Software Architecture.

The public ASPIRE Reference Architecture document is an important S&T results because it helps users of the ASPIRE S&T results and their prototype implementations to understand and use those artefacts, and because it helps them to reason about the protection provided by the ASPIRE protections and combinations thereof. It also provides an excellent framework for reasoning about future extensions and compositions.

1.3.1.2 Task T1.5 Validation

One of the most important S&T results of the project is deliverable **D1.06 Validation**, a **public 66-page** report in which all of the developed software protection technology is validated on the three project use cases, against the security requirements, and in light of the attacks of the attack model. In a preliminary version D1.05 delivered after two of the three project years, the members of the ASPIRE advisory boards confirmed that the validation methodology was fair and complete.

In D1.06, the ASPIRE consortium concluded that the assembled (and updated) **Attack Model, Security Requirements, and Reference Architecture were still valid and appropriate** at the end of the project. The consortium also concluded that the **project results meet almost all requirements**. In particular, the most important ones are met. In addition, the ASPIRE consortium concluded that, with the exception of some data obfuscations and multi-threaded cryptography, all protections developed during the project are covered by the validation of the use cases.

Regarding the assets in the ASPIRE use cases and their security requirements, the consortium concluded that at the end of the project the **envisioned, useful protections for those assets are mostly available in the ASPIRE protection framework**. The exceptions were interprocedural data obfuscation (which is not available), control flow tagging (which was delivered too late by one partner to protect another partner's use case), and dynamic remote attestation (which was not yet supported on the use case demonstration platform). Moreover, the ASPIRE consortium concluded that the **tool support for the available protections was sufficiently mature** to deploy them correctly and automatically on the complex project use cases.

Based on their own assessment and tests, as well as on penetration tests performed with project outsiders, i.e., in tests performed by academic students, professional tiger teams, and in a public challenge, the ASPIRE consortium concluded that although many of developed and integrated protections still offer a large potential for improvement, those **protections effectively delay attacks and increase the effort that attackers need to invest in identifying attack vectors**; **they make it harder to exploit identified attacks at a large scale; and hence effectively reduce the profitability of engineered attacks.**

It also concluded that it can conjecture that the developed and integrated **renewability techniques deliver improved protection**, as they make the scaling up of attacks harder, as they can delay attack vector identification, and as they can help in raising the costs faced by attackers.

To a large degree, the project has hence achieved its goals to demonstrate that software-based protection techniques that cover multiple lines of defence can be composed to deliver true protection.

Relying on various experimental evaluations and theoretical considerations, the ASPIRE consortium furthermore concluded that **the overhead the protections introduce is limited and can be controlled** by deploying the protections cautiously. In particular, on the project use cases, the protections could be deployed as foreseen by their developers and the project's security experts, and with an acceptable overhead.

Finally, the ASPIRE consortium concluded that the **ASPIRE Decision Support System has a very high potential** even if it is not yet ready to be used to protect real applications. On one hand, it can **automate complex operations** relevant to the deployment of software protections, such as attack path discovery and suggest combinations of protections to deploy; it gathers a huge amount of data useful for making decisions; and it proposes effective protections. On the other hand, in some cases, the level of details of the output is not yet good enough to be used in practice (attack paths description are too coarse grained); it was not possible to prove that the golden combinations are actually optimal; and there are still concerns about the presentation of the results, especially with the consulted experts outside the ASPIRE project.

To a large degree, the project hence achieved its goals to demonstrate that decision support can ease and semi-automate the task of the user of a software protection tool chain. Tool support is available that, given annotations of assets and a description of the available protections, their limitations and their impact on security and overhead, can provide insights in the best available combinations of protections to deploy, and provide quantitative evaluations of them.

1.3.2 WP2 Offline Software Protections (M01-M30)

The results of WP2 were documented extensively in deliverables D2.01, D2.03, D2.06, D2.08, and D2.10, all of which are public. The developed prototype tool support was delivered in D2.02, D2.04, D2.05, D2.07, and D2.09, most of which have been open-sourced.

1.3.2.1 Task T2.1 Data obfuscation

At the beginning of the project, FBK surveyed the state of the art relevant for data obfuscation and identified the most interesting approaches. Some promising approaches have been implemented to experiment with their strong and weak points. These initial findings have been published as a workshop paper, in the 1st International Workshop on Software PROtection (SPRO), held in May 19, 2015, in Florence, Italy.

One of the most interesting among the existing approaches relied on an NP-hard problem to block attacks. The rationale is that an attacking tool should have solved a hard problem to recover clear values. However, we realised that this approach was limited by the fact that the same hard problem that should stop the attacker also limits the obfuscation tool. In fact, the protection tool should check the solution of a randomly generated hard problem, by solving it at obfuscation time. Consequently, to make the obfuscation process affordable, either the NP-hard problem should not be too hard or the transformation soundness should not be checked at obfuscation time. The Authors of this approach adopted the second strategy to deliver a hard obfuscation scheme that might deliver wrong obfuscated code. As generating potentially wrong obfuscated code is not acceptable in the scope of ASPIRE, this approach needed to be refined and extended to meet the project requirements.

FBK therefore invented a novel approach to data obfuscation that overcomes this limitation by using a mapping between different NP-hard problems. At obfuscation time, a tool generates a (random) instance of a 3SAT problem that is fast to check by the obfuscation tool. This ensures that the transformation is sound. This easier instance of a 3SAT problem is then transformed into a larger instance of k-clique problem that is much harder to solve. The transformation algorithm then obfuscates data such that a static de-obfuscation attack tool would need to solve the hard k-clique problem to recover clear data.

The experiments confirmed that:

- A state-of-the-art static analysis tool requires exponential time to recover clear values of obfuscated variables;
- The obfuscation transformation requires very short time to protect values; and
- The transformation is sound.

These research outcomes have been accepted for publication in an academic conference, the 24th IEEE International Conference on Software Analysis, Evolution, and Reengineering, to be held in February 20-24, 2017, in Klagenfurt, Austria.

Automated tool support to deploy the implementations of the existing state of the art and of the invented extensions has been delivered and integrated into the ASPIRE Compiler Tool Chain. This includes support to extract source code annotations that specify the protections to be deployed.

The project goals of integrating state-of-the-art data obfuscations, of pushing that state of the art, and of developing automated tool support have hence clearly been achieved.

1.3.2.2 Task T2.2 White-box cryptography

White-box cryptography

In the state of the art, prior to ASPIRE, there was no research into provably secure constructions. This was something that NAGRA wanted to find out, and it chose to explore how multi-variate white-box constructions can be achieved. The activities in the ASPIRE project showed that such provably secure constructions can indeed be achieved. NAGRA found ways to do so, but concluded that these constructions are too slow and too large for practical use cases.

For the remainder of the white-box activities in the ASPIRE project, NAGRA focused on practical implementation and improvements on the white-box tool framework for ASPIRE (WBTA). The WBTA framework is based on NAGRA's existing tools for managing the white-box implementation generation and build process. It was adapted (i) to allow the integration of new white-box constructions that were developed during the ASPIRE project, and (ii) to integrate it into the ASPIRE Compiler Tool Chain and to make the constructions composable with many other protection techniques – this was one of the prime goals in the project. The WBTA is integrated into the tool chain as a source-to-source translation tool that operates on annotated code and rewrites that code to include invocations to newly generated white-box code. That white-box code is generated automatically by the WBTA and linked into the program. As a result of the integration into the ASPIRE Compiler Tool Chain, the translated code and newly generated white-box code can be protected with other binary-level protections and with some source-level protections such as cross-translating it to bytecode, applying control flow obfuscation, applying integrity checks, etc.

NAGRA also investigated new constructions for fixed-key implementations, which have been deployed onto its use-case, and found new ways to achieve dynamic-key implementations. Such white-box implementations have not been seen in public literature before, and allow to instantiate keys in white-box implementations at run time – something that is needed in DRM use cases. NAGRA also developed faster, time-limited white-box implementations which introduce a trade-off between security and performance, and then subsequently investigated how such implementations can be renewed at run time (See Task T3.3 in Section 0). Such time-limited implementations can be useful in bootstrapping use cases or other use cases where cryptographic keys have only value in a limited time-frame.

The practical implementations were deployed successfully on two of the three industrial project use cases through the ASPIRE Compiler Tool Chain, and not broken in any of the multiple penetration tests in which they were attacked.

Together with SFNT, NAGRA also researched options for using WBC to protect VM bytecode. The conclusion from this is that apart from encrypting the bytecode – which is a straightforward process not very useful to invest research funding in – nothing useful enough and with sufficiently low overhead was found.

With respect to pushing the state of the art of white-box cryptography and the integration of tool support for practical instantiations, the project has as a result achieved all its objectives.

Diversified Cryptographic Library

GTO proposed a protection that enables to hide a key at compile time in the application in such a way that it can be derived in a secure way at run time. A typical use case is a symmetrical master key that is shared between the application and a server from which derived keys are computed with data retrieved from a service provider server. The Diversified Crypto library (DCL) protection enables such a feature. This protection is derived from other research work done at GTO and enables to embed some application code together with the pure cryptographic library. In this regard is it more than another cryptographic library available for mobile application because it brings flexibility.

For the purpose of the validation task, the one-time password (OTP) generation processing (see the OTP use case in Section 0) has been put within the DCL. In addition to protecting a master key as initially planned, this protection can deploy application logic as well and take advantage of the secure storage feature of DCL and the ability to embed some application code under an encrypted form.

With the development of the DCL, the integration of its tool support into the ASPIRE Compiler Tool Chain, and its deployment on the OTP use case, the project has reached the goal put forward in the DoW for this protection.

1.3.2.3 Task T2.3 Client-side code splitting

In the DoW, SFNT was foreseen to develop a protection and the corresponding tool support called client-side code splitting. In this protection, native code is extracted from the code section of the binary/library to be protected. The native code is then cross-translated to bytecode, which is injected into the binary or library, together with a software virtual machine (SoftVM) that can interpret the bytecode. SFNT delivered three generations of SoftVMs with the corresponding cross-translators, albeit as Background. Because of the importance for SFNT's business, they already developed SoftVMs meeting the requirements of their business ahead of the project schedule. In the project, those SoftVMs were re-used rather than developing alternative ones from scratch. Whereas the first two generations still contained weaknesses (as expected by the developers and confirmed in the conducted penetration tests), the last one is considered significantly stronger because it is not vulnerable to the attack steps that were executed during the penetration test.

A tool flow design for the code extraction, cross-translation, and injection of bytecode was designed and developed by UGent and SFNT (as project Foreground) to enable the deployment of this protection as a plug-in in the overall ASPIRE Compiler Tool Chain. Its main feature is a clear separation of concerns between the native code extraction and bytecode injection functionality on the one side, of which the only feasible implementation is one that is tightly coupled to the other binary-level protection tool support, and the cross-translation and SoftVM compilation on the other side, both of which should be able to evolve and be replaced independently of all other protections. Still both sides of the separation co-operate tightly, because the extraction tool queries the cross-translation tool to identify which native instructions it supports.

The native code extraction and the bytecode injection functionality are open-sourced. Finally, it is worth mentioning that the designed tool flow makes that this protection composes really well with other protections. In particular, the linked-in SoftVM can be protected with all binary-level protections, including binary code obfuscation, code guards, migration to the self-debugger, etc. Furthermore, code from the SoftVM as well as the translated bytecode fragments can be made mobile.

Whereas the DoW proposed to research techniques to automatically identify the code fragments to be cross-translated with slicing techniques, external expert advice gathered during the project (from advisory boards and other experts of the industrial partners) made us move away from this idea. The advice in general was to focus on techniques in which the protection tool chain user explicitly marks code to be protected (i.e., annotates the code fragments relevant for protecting the assets), thus avoiding the in general intractable problem of automatically determining that code based on a description of only the assets. For this protection, we completely followed that advice. This decision followed in part from the fact that slicing techniques to identify relevant code were already being researched for client-server code splitting later in the project anyway (in Task T3.1, see Section 1.3.3.1), and that little additional scientific value was to be found researching it earlier here and hence delaying the other work in this task until a slicing technique would become available.

Overall, the project has hence clearly met all its goals with respect to the client-side code splitting protection in this task.

1.3.2.4 Task T2.4 Binary code obfuscation

Starting from UGent's embryonic Intel x86 support for code flattening, opaque predicates, branch functions, and code layout randomization in its link-time binary rewriting framework Diablo, UGent developed a generic Diablo control flow obfuscation module, and back-ends for both the x86 (i386) architecture and the target ARMv7 platform of this project. In the penetration tests conducted in the project, these obfuscations proved to hamper attackers, but they were not (yet) unsurmountable. This is okay, as the goal of this effort was to enable integration of state-of-the-art binary code obfuscations with other protections, not to push the state of the art w.r.t. code obfuscations.

That integration goal was clearly reached: in the binary-level processing step in the ASPIRE Compiler Tool Chain, the code obfuscations compose well with the other protections. All components linked into the binary/library in support of other protections (the ASPIRE communication logic, the SoftVM, implementations of WBC primitives, code guards, the self-debugger, ...) can be obfuscated. Moreover, the more global obfuscations (flattening, branch functions) as well as the global transformation of code factoring can be applied across functions, even across functions from different components (linked-in object files or archives), and can hence hide the boundaries between the components. Furthermore, the code that is extracted to become mobile, can be obfuscated as well. The project goals with respect to generic control flow obfuscation have hence clearly been reached by UGent. Importantly, all of the developed and integrated tool support has been open-sourced.

In a parallel research track, GTO designed a multi-threaded cryptography scheme to provide domain-specific obfuscation for cryptographic primitives in online client applications. As this scheme seemed promising at first, a prototype implementation was developed and evaluated. At that point, it was observed that in order to deploy the technique, the APIs (input/output formats) of the client need to be adapted. Automated rewriting of such APIs falls outside the scope of the project, however, so the further development within the project was stopped.

With respect to code obfuscations, it is clear that the project achieved all objectives.

1.3.2.5 Task T2.5 Anti-tampering

UGent designed a tightly coupled self-debugger protection in which a custom debugger occupies the one seat available in operating systems for debugging processes under attack. It works by forking off a debugger process as soon as a binary or library is loaded into memory, and by letting that debugger attach to all the threads in the process to be protected. This prevents an attacker from attaching his own debugger. This proved to be a very effective anti-debugging protection in the conducted penetration tests. Moreover, it prevents the collection of execution traces with existing system-level instrumentation tools on the project's target platform. UGent designed not only the overall protection technique, but also an implementation that operates correctly in the very demanding scenario of the project use cases, incl. one case in which a library to be protected is loaded and unloaded repeatedly into a multi-threaded daemon server process. Furthermore, UGent developed the necessary tool support in its Diablo link-time rewriting framework (and integrated it into the ASPIRE Compiler Tool Chain), such that the deployment of this protection is fully automated.

All tool support and protection components will be open sourced (for research purposes). The work was also published in an academic workshop, the Software Security, Protection, and Reverse Engineering Workshop that was co-located with the ACSAC 2016 conference on 5-6 December 2016. Finally, to assess the exploitability of this technique, which depends on the availability of a number of kernel functionalities on the device platform, the protection was tested on the latest Android and Linux versions. It worked without problems.

UGent also developed anti-callback stack checks. The developed tool support can inject these checks automatically in all non-exported functions of binaries and libraries, where they check for the occurrence of unallowed external function calls every time such a function is invoked. The checks are light-weight, checking only for coarse-grained control flow integrity, because they need to operate in an environment (i.e., they are loaded into a process space) not controlled by the ASPIRE Framework, but by Google (i.e., the Dalvik runtime environment and the DRM server and media server daemons). The tool support was integrated into the binary-level processing steps of the ASPIRE Compiler Tool Chain and open sourced and it functions correctly on the project use cases. During the professional penetration tests in the project, no attacks on these checks were executed, however.

Furthermore, UGent designed and developed a protection consisting of offline code guards, and the necessary tool support to deploy them automatically. The developed tool support manipulates source code to inject invocations of attestators and verifiers at locations indicated by source code annotations. Based on the forms of guards requested in the annotations, source code implementing the attestators and the verifiers is then instantiated from a template, together with a set of application-independent, to a certain degree stealthy reaction mechanisms, and a set of data structures in which the verifiers can encode, in non-obvious ways, whether or not their verifications were successful. The reaction mechanisms are carefully engineered code fragments that can be executed at any point in time; they will work correctly when all verifications so far were successful, but cause havoc in unsuspicious ways if not, e.g., by causing corruption by means of free-after-free, and causing deadlocks by means of locking primitives) if not. Those are then compiled and linked in into the protected program. In the binary-level processing step, invocations to the reaction mechanisms are then injected throughout the protected application, and data structures are populated and linked in into the binary that describe the (randomized) layout of the regions to be attested in the final binary. Those data structures are then consumed at run time by the attestators. All tool support for this protection was integrated into the ASPIRE Compiler Tool Chain, and has been open sourced.

The offline code guard protection composes well with the other protections: the code implementing the attestators, verifiers, and reactions in the protected application can be made mobile, it can be moved into the self-debugger, it can be obfuscated, it can be cross-translated to bytecode, etc. The protection can guard all code in the protected binary, incl. the original application code, the SoftVM, the debugger, the communication logic, the guard code itself and other anti-tampering checks. In that respect, the project goal of composing the existing state of the art in code guards with many other protections was certainly reached.

On small programs, as in the public bounty, this protection proved to be vulnerable because when it is deployed on few program locations, it can be undone manually. Still, this protection delayed the attacks on the public bounty challenges significantly. Moreover, manual undoing of the protection does not scale to larger programs.

A minor objective of this task in the DoW was the *integration* of *pre-existing* anti-code-injection techniques such as technique that prevent the inject of additional threads by checking that only valid, expected threads are present. This objective was abandoned based on three considerations. First, no existing protections were found that actually can do this for dynamically linked libraries. Secondly, and obviously in line with the first reason, the consortium judged that designing and developing such a protection for dynamically linked libraries such as the project use cases, which are loaded into third-party applications of which the threading behavior is unknown and not limited in scope, is extremely challenging, and that devoting resources to this topic is hence introducing a high risk. Finally, the consortium judged that, as the goal of this minor foreseen effort was to study the integration of yet one more technique into the protection tool chain, and not the development of a new state-of-the-art-pushing anti-code-injection techniques; and as many techniques are already integrated and demonstrated to be truly composable, the added value of yet another technique would be very small, and hence not worth the mentioned risk.

Overall, it is therefore safe to claim that with respect to offline anti-tampering protection techniques, the project achieved all major goals.

1.3.3 WP3 Online Software Protections (M07-M36)

The results of the research conducted in WP3 were presented in detail in deliverables D3.01, D3.04, D3.06, D3.08, and D3.09, all of which are public. The software prototypes were delivered in D3.02, D3.03, D3.05, and D3.07.

1.3.3.1 Task T3.1 Client-server code and data splitting

Client-server code and data splitting

Client-server code and data splitting was a challenging protection, because it required to meet several contrasting requirements:

- Moving a significant portion of code to the server, such that attacker comprehension is reduced and tampering opportunities are drastically reduces;
- Moving a portion of code small enough to control server overhead (CPU time and memory consumption);
- Minimizing communication between client and server, to control network overhead and to control the delay that could potentially impact the user experience.

To meet all these requirements, FBK developed a novel splitting strategy based on program slicing, i.e., to follow control and data dependencies and split a logically contiguous portion of program. A prototype developed during the first year of the project highlighted a critical problem, related to the transitive inclusion of dependencies in the slice: The initial slicing algorithm included too many library functions. This caused compatibility issues between the program to protect (that uses ARM libraries) and the slice to be run on the ASPIRE server (that uses x86 libraries). During the second year, FBK elaborated a solution to this problem and refined the splitting algorithm into an improved solution that can reduce of the size of the slice (i) to avoid the inclusion of the body of library functions in the slice and; (ii) to generate a smaller slice to be executed at server side, thus reducing the server load.

It is important to note that whereas the DoW treated code splitting and data splitting as related, but different techniques (data splitting requiring extensions to code splitting) the developed approach actually inherently combines both, as both the data on which the slices are computed, and the code in the slices are moved onto the server.

These research achievements have been implemented in a fully automatic transformation tool, that was subject to extensive validation, including of the performance cost of the transformation on a set of case studies in terms of execution time overhead, the memory overhead, and the network overhead for several configurations of the protection. This tool will be further developed and commercialized in the spin-off that FBK is initiating.

This protection was also subjected to extensive validation from the security point of view, as it was studied in isolation (i.e., combined with no other ASPIRE protection) in the second round of experiments with academic participants, and as it has been subjected to experimental validation combined with other ASPIRE protections in the experiments with industrial tiger teams and in the public challenge.

Finally, this protection was combined with remote attestation: when tampering is detected with remote attestation, the server stops responding to the client, which can hence not continue executing. This reactive remote attestation technique published in a peer-reviewed paper: Alessio Viticchié, Cataldo Basile, Andrea Avancini, Mariano Ceccato, Bert Abrath, Bart Coppens. Reactive attestation. Automatic detection and reaction to software tampering attacks. In Proceedings of the 2016 ACM Workshop on Software PROtection (SPRO 2016), pages 91-92, 2016, ACM.

With respect to client-server code and data splitting, the project hence reached all of its goals.

Code and Data Mobility

At the beginning of the project, UEL brought its background knowledge of software protection based on mobile code delivered from a trusted network server. The initial Windows-based prototype has been completely re-designed and re-implemented by UEL and UGent into a novel approach for Code Mobility, relying on binary rewriting features of the Diablo framework, so that the binary code transformation is correct and equivalent to the original ARM binary code.

Once the binary has been compiled and protected, the developer can use code mobility to choose which functions or parts thereof will be transformed into mobile code blocks to be downloaded from a secure server at run time as soon as they will be invoked for the first time in the client application.

The percentage of code to be made mobile and the different network configurations have been tried in many experiments, which have been documented in a published paper: Alessandro Cabutto, Paolo Falcarin, Bert Abrath, Bart Coppens, Bjorn De Sutter. Software Protection with Code Mobility, In Moving Target Defense International workshop (MTD-15), co-located with CCS-2015, ACM.

UGent designed and developed the necessary extensions to support not only mobile code, but also mobile data, which has been a key enabler to support dynamic white-box cryptography (in which tables encoding new or re-randomized keys are made mobile) and SoftVM bytecode mobility, in which case the bytecode is not stored statically in the client binary but downloaded from the secure server as well.

The code and data mobility support code has been already exploited within the ASPIRE project as transport layer for mobile code blocks in the renewability framework (discussed later in Section 0), and for remote attestation support (denying mobile blocks delivery in case of tampering detection). The code has been open sourced and published on GitHub, and it has been integrated into the ASPIRE Compiler Tool Chain.

This protection was also subjected to extensive validation from the security point of view, as it has been subjected to experimental validation combined with other ASPIRE protections in the experiments with industrial tiger teams and in the public challenge.

Finally, is worth noting that the components (Binder and Downloader) injected into a protected application to deploy the mobility can be protected with all other binary-level protections of the ASPIRE Compiler Tool Chain. Here again, a high level of composability has been achieved.

With respect to code and data mobility, the project has exceeded the original expectations from the DoW: not only can bytecode be delivered from a server, but so can native code. This is a great achievement, because it greatly improves the composability of code and data mobility with other techniques.

Concretely, this achievement makes other techniques renewable without requiring the presence of a SoftVM. Renewability can therefore be implemented with less overhead, and is made available to users that do not have the necessary access rights for the client-side code splitting protection, such as users of the open-sourced prototypes.

ASPIRE Common Communication Logic

All online techniques owners successfully integrated client and server components using the ASPIRE Server-side Communication Logic and the ASPIRE Client-side Communication Logic ASCL libraries developed and provided by UEL. Together, these form a shared layer for bi-directional communications between clients and server used in all the online protections. Such logic has been widely tested and validated against several scenarios including teams experiments with industrial tiger teams, the public challenge, industrial use cases demos, and internal tests.

The initial design has been extended, fully implemented and deeply tested to meet ASPIRE partners' needs. The result of this work is a robust, reusable, easy to implement client-server support library based on well-known open source projects (curl library) and standardized protocols, such as HTTP and WebSockets. The prototype implementations have been open-sourced.

Finally, the libraries can be protected with all binary-level protections of the ASPIRE Compiler Tool Chain, thus achieving high composability.

With this achievement, ASPIRE eases the take-up of the ASPIRE open sourced results, as it will greatly reduce the effort needed by future users to integrate new online protections.

1.3.3.2 Task T3.2 Remote attestation

Remote Attestation

Attesting the integrity of a running application is a complex task; performing it without secure hardware, as for several mobile devices and embedded systems, is even more challenging. Taking advantage of a trusted remote server to determine when and what to attest, to emit trustworthiness verdicts and decide reactions, remote attestation (RA) gives the possibility of a (theoretically) higher protection compared to embedded software guards. However, it introduces several design and performance constraints we had to face in the ASPIRE project.

While in literature several works exist that define the design of remote attestation techniques, in practice no open source tools were available to download, adapt and use in the ASPIRE project. The Consortium then decided to invest on implementing RA techniques to complete the five lines of defense and, given the increased importance of the automatic tool chain, to automate the RA application on target application, and integrate RA into the ASPIRE Compiler Tool Chain.

We have selected a form of static remote attestation (Static RA) that attests the integrity of applications by checking that the binaries in memory have not been altered, and a dynamic form Remote Attestation (DynRA) that monitors the correct execution of an application based on (likely and true) invariants. Automated tool support has been developed for both the techniques. Moreover, static RA has been integrated into the ASPIRE Compiler Tool Chain and open sourced (https://github.com/aspire-fp7/remote-attestation), and applied to the project use cases, as well as several open source applications. DynRA has been tested on open source application on x86 platforms due to limitations of existing likely invariants extractors on the ARM platform for which the project use cases were developed. To push the state of the art, we have investigated several interesting research directions for Implicit Remote Attestation, a variant of RA that obliviously collects the information useful to decree about the trustworthiness of an application without the need for a client-side attestator explicitly sending evidences. Unfortunately, some advanced aspects of Implicit RA that were investigated, lead to negative results and were not implemented because of incompatibilities with the use cases and other protections.

The RA server side support has been designed to manage several instances of the same application and several applications at the same time, independently of the actual remote attestation. All the server-side components are independent by design and only interact through a database. The RA-specific verifiers are dynamically instantiated when applications require them, and may be replicated and executed on different machines for load balancing purposes. Indeed, the server scalability is one main issue for companies that may have to face critical periods when thou-sands or millions of clients may be connected at the same time (e.g., streaming the world cup final). Moreover, pre-computations of verification tasks are performed offline to mitigate this risk.

A framework for reactions has been implemented, where reaction strategies are defined with policies (which may be driven by business and security considerations) and can be individually associated to single instances of applications. ASPIRE has produced server-driven local reactions, that render unusable applications when the server order them (time bombs). Moreover, we have pushed the state of the art with the concepts of positive attestation, i.e., applications are rendered unusable if they do not prove their integrity (and punished when fail), and delay data structures, i.e., covert channels that need to be created between server-side reaction logic and local reaction techniques. Moreover, server side reactions have been developed. The Reactive Attestation instruments applications to make them not executable without interacting with a server, which is notified by the reaction logic and stops serving compromised applications. Analogously, applications that are also protected with code mobility can be rendered unusable by notifying and stopping pushing updated code blocks.

As already mentioned in Section 1.3.3.1, one form of reactive attestation has already been published, i.e., the form where remote attestation was combined with client-server code splitting.

Finally, it is worth pointing out the integrated forms of RA compose well with all other protections. For example, fragments in the injected software components that implement static RA and dynamic RA can be cross-translated to bytecode, they can be made mobile, they can be obfuscated, etc.

In conclusion, with the composable designs and implementations of static RA and dynamic RA, and with the novel server-side management logic that pushes the state of the art, we have achieved the project goals with respect to the fifth line of defense, remote attestation. We also explored more advanced forms of RA, but more research is still needed, on a broader set of use cases.

Anti-cloning

NAGRA has developed and integrated an anti-cloning protection. This technique allows to detect at the server-side if identical copies (clones) are executed on clients. The idea of the technique is to monitor the state of the application, and dynamically update this state as the application is interacting with a server. Since identical copies will interact independently with the server, their state will become de-synchronised, which can be detected by that server. In contrast to most techniques in ASPIRE (which focus on mitigating reverse engineering or tampering), this technique focuses on mitigating the large-scale exploitation of attacks. The technique itself needs to be further protected to avoid that copies can easily re-synchronize, and as such, this technique has been further composed with other protection techniques in the ASPIRE project.

The anti-cloning technique is a source-level protection technique. Support has been implemented for integrating the anti-cloning library during the build process of a client-side application by the ASPIRE Compiler Tool Chain, as well as server-side support for keeping track of client applications and potential clones. This technique is able to detect if applications have been cloned, and will track them such that (based on a pre-defined policy) action can be taken. An additional server-side interface as implemented such that other online protection techniques or the original application service can take appropriate action. For example, the mobile code technique can query this interface and decide not to send new mobile blocks to the client application when it has been flagged as being a cloned application. Or a streaming movie service can refuse to stream content to an application when this has been flagged.

The project has a such achieved all objectives of the DoW with respect to anti-cloning.

Control Flow Tagging and Reaction Mechanisms (a.k.a. Software time Bombs)

GTO has developed the Control Flow Tagging (CFT) protection. It is an anti-tampering protection to detect an alteration of the binary code at run time. The initial idea was to combine a technique used in code coverage tools and a self-degradation of the application as reaction mechanism in case the application logic has been tampered with.

It happened to be, however, that other techniques could use the reaction mechanism, so it has been decided in year 1 that the initial protection should be split in two distinct techniques.

- The CFT protection sets Gates in the binary code on the application where a numeric counter check the number of time the activation has passed the Gate. Logical expressions can control that Gate counters have expected values at
- The reaction part called the Software Time Bombs in the DoW and generalized in the Reaction Mechanism in the deliverables.

At compile time the logical expressions that combine the Gate counter values are put apart to be placed on the ASPIRE server under binary format to enable the Reaction Mechanism to evaluates the logical expressions. A logical expression combines Gate counters only with comparison operators and logical connectors, so this code can be compiled and executed remotely on the server provided the Gate counter values are sent to the server.

Note that both the CFT and the Reaction Mechanism have also been implemented as offline techniques. In this case the logical expression is evaluated locally in the application and if not satisfied the reaction is triggered. The delay effect of the reaction is important to prevent the attacker to connect the Gates, the logical expression evaluation and the actual degradation due to the Reaction Mechanism.

All developed tool support was integrated int he ASPIRE Compiler Tool Chain and deployed on GTO's use case.

By broadening the initially foreseen designs and supporting both offline and online versions of the control flow tagging protection and the reaction mechanisms, the project has more than achieved its goals with respect to these anti-tampering protection components.

1.3.3.3 Task T3.3 Renewability

Code and data mobility provides excellent protection against static analysis attacks but once dynamic or hybrid techniques (such as taking memory dumps and analysing those) have succeeded, the offer little protection. The Renewability Framework designed and developed within the ASPIRE project overcomes this limitation by providing renewable and diversified mobile code blocks. Support for this framework was integrated into the ASPIRE tools and all prototype implementations are open-sourced.

UGent put its effort on extending Diablo so that is possible to obtain diversified mobile code blocks. UEL designed and implemented the logic that invokes such support tool when necessary and delivers it to the client for deployment, depending on various configurable renewability policies. The client hence receives different code every time it is executed, which clearly is a form of renewability in time. This necessary tool support, incl. script generation tools of which the produced scripts can be run on a server to generate additional diversified blocks, has been integrated into the ASPIRE Compiler Tool Chain and has been open-sourced.

SFNT contributed its diversifiable SoftVM and the corresponding diversifying cross-translator (as Background) to make the client-side code splitting approach renewable in space: each software instance can have a custom bytecode and interpreter to limit the attacker's learnability of the client-side code splitting protection. This is clearly a form of renewability in space: different customers can get different versions. It can also be deployed as a form of renewability in time, however, when successive releases of a software product feature diversified bytecode and SoftVM. SFNT and UGent integrated the tool support for this form of renewability into the ASPIRE Compiler Tool Chain.

NAGRA, UGent, and UEL worked on white-box crypto renewability, such that by delivering re-randomized keys by combining code and data mobility, attackers can only reuse broken keys for very short time frames. The necessary tool support was developed and integrated into the ASPIRE Compiler Tool Chain. This includes script generation tools of which the produced scripts can be run on a server to generate the mobile blocks that implement new or re-randomized keys. This is clearly a renewability-in-time protection.

Static, native code diversity in space was also researched: UGent generalized its existing framework for static native code diversification in and on top of its Diablo link-time rewriter, and developed implemented support for the target platform of the project (ARMv7 Android). The tool support was integrated into the ASPIRE Compiler Tool Chain and is part of the open-sourced prototypes. The deployment heuristics were also improved, as was the controllability by means of source code annotations.

Furthermore, UGent focused on making diversification more feasible in practice. On challenge is that when diversified applications are delivered to end users, collecting accurate and usable crash reports is problematic because each collected stack dump implements a different layout (of both the stack and the code segment of the crashed binary). UGent designed and prototyped an extension of the widely used Breakpad crash reporting tool (from Google) that stores a minimal amount of additional information in the diversified binary and sends it to the crash server together with the stack dump. With the additional information, the crash server can then reconstruct a human-readable stack trace, without having to store debug information on the crashed, diversified software instance. Once the paper describing these results has been accepted for publication, the prototype tool support will be open sourced.

Finally, UEL and FBK researched how to obtain maximally diversified application copies. They conducted that research work on diversity in space applied to Java code. They applied NCD metrics to measure diversity between diversified versions of the same application, clustered such versions to obtain maximally diversified versions; this work has been published in a conference paper: Mariano Ceccato, Paolo Falcarin, Alessandro Cabutto, Yosief Weldezghi Frezghi, Cristian-Alexandru Staicu. Search Based Clustering for Protecting Software with Diversified Updates. In Symposium on Search-Based Software Engineering SSBSE-2016. Springer, pp 159-175.

With these achievements, the project certainly met all its goals with respect to the fifth line of defense, renewability.

1.3.4 WP4 Security Evaluation (M04-M36)

All results obtained in WP4 are presented in the extensive deliverables D4.01, D4.02, D4.03, D4.05, and D4.06, all of which are public.

1.3.4.1 Task T4.1 Security Model and Evaluation Methodology

Models and Knowledge Base

POLITO, UEL, and UGent contributed to the design of the ASPIRE Knowledge Base (AKB) to describe statically a conceptual model needed by the ASPIRE Decision Support System (ADSS) to perform its tasks.

Modelling all the concepts and properly relating them was a foundational step to build all the evaluation methods and the decision support. The Consortium has thus developed the ASPIRE Security Model and continuously maintained it until the last version (ASMv1.2).

The ASPIRE security model depicts both a priori general knowledge about the software protection scenario and relevant information about the applications to protect. These data are used to infer additional knowledge by means of the internal inferences of the ASPIRE Knowledge Base (AKB, which is a OWL-DL ontology) and ad hoc enrichment modules (which perform custom reasoning and inferences) coordinated by an enrichment framework, directly driven by the ADSS to perform its risk analysis.

The security model is formed by a main model, which describes and the high-level concepts and the relationships among them, and a set of sub-models (Application, Asset, Attack, Attack path, Attacker, Software protection, Tools), which refine the concepts in the main model to increase reasoning precision.

The ASMv1.2 is provided as a set of UML Class Diagram models, represented in XML and valiadated with an ad hoc XMLSchema. Moreover, the ASMv1.2 is also available with the AKB ontology. The ASMv1.2 and the related documentation has been open sourced with the ADSS (available here https://github.com/SPDSS/adss).

Evaluation Methodology and ASPIRE Decision Support System-Light

UEL developed the ADSS-Light, a tool that help the software developer in manually modelling the possible attacks to the application assets with Petri Nets and in assessing the strength of the protected program by evaluating combinations of binary code metrics.

In our case a Petri Net represents a set of attack paths that the attacker can perform to achieve their goal on one or more assets in the application: each path is a sequence of attack steps, which corresponds to transitions in a Petri Net.

The ADSS-Light aims at helping expert software developers with a semi-automated approach to assess the security strength (Protection Fitness) of a protection instantiation (i.e., a configuration of a set of protections in the ACTC). The user has to perform some initial configuration steps, such as associating binary code metrics and code regions to attack steps, customizing the metrics formulas, and finally choosing the type of assessment.

The ADSS-Light has been applied to the NAGRA use case and it relies on metrics calculated by the metrics framework in the ACTC.

In case of missing data and metrics for some attack steps, these data can be set as random variables and use to run a Petri Net simulator, whose initial results have been published in a conference, and received the **best paper award**: Gaofeng Zhang, Paolo Falcarin, Elena Gómez-Martínez, Shareeful Islam, Christophe Tartary, Bjorn De Sutter, Jerome D'Annoville. Attack simulation based software protection assessment method. In International Conference On Cyber Security and Protection of Digital Services (Cyber Security 2016), Pages 1-8, IEEE, (Best paper Award).

A journal extension in going to be published soon: Gaofeng Zhang, Paolo Falcarin, Elena Gómez-Martínez, Shareeful Islam, Christophe Tartary, Bjorn De Sutter, Jerome D'Annoville. Attack Simulation based Software Protection Assessment Method with Petri Net. In International Journal on Cyber Situation-al Awareness, Vol 1:1, ISSN 2057-2182 2016 (in press).

The ADSS-Light and the underlying software components have been open-sourced.

With the developed, formalized models; the knowledge base and its tuned content; the security evaluation methodology and the tool support to deploy the methodology, the project has achieved all goals of this task.

1.3.4.2 Task T4.2 Complexity metrics

First, the consortium proposed a comprehensive software metric framework that comprises a wide range of **software complexity metrics** to cover the relevant software features and representations (e.g., static representations such as control flow graphs, and dynamic representations such as program traces and dynamic program slices) for a wide range of attack steps (i.e., the steps in the ASPIRE Attack Model). For modelling the effort an attacker needs to invest in a particular attack step on a particular code fragment, i.e., the time he will need and hence the delay towards reaching his end goal), a specific combination of metrics is to be computed and aggregated, with a formula that depends on the features of the specific attack step and the specific goal the attacker is trying to achieve with that step. With the software complexity metrics, the **potency** of protections can be computed. Moreover, some of the complexity metrics center on variability aspects when comparing multiple traces of multiple versions of a protected program, and thus provide a mechanism to quantify important aspects of a protection's **stealth**.

Around the start of this task in M04 of the project, the consortium became aware of the new line of research that was being conducted by Prof. Debray and his students at the University of Arizona (USA). They developed a new approach to **automated, generic de-obfuscation** based on the simplification of program traces. The simplification is based on information about the **semantic relevance** of operations occurring in the traces, and on the **variability** observed on the operations' operands. The consortium (in particular the coordinator, who had previously collaborated intensely with Prof. Debray) recognized that these simplifications to a large extent match with the heuristics/simplifications that attackers deploy to simplify their mental models (i.e., comprehension) of software under attack, and that it also was a generalization of existing code simplification techniques as deployed by attackers while they tamper with code to remove protections such as opaque predicates or code guards.

From that starting point, we then further generalized the approach to let it cover all the ASPIRE protections and a wide range of possible attacks thereon. Building on the concepts of semantic relevance and variability, a range of **resilience metrics** were then proposed. With these resilience metrics, the aforementioned computation of the complexity metrics is parametrized (i.e., weighted) for each attack step, such that the computed potency takes into account the mental heuristics or executed attack scripts already deployed beforehand by an attacker to minimize his effort and hence his delay in the identification and engineering of an attack.

Finally, we proposed a number of so-called **unavailability metrics** to measure the impact on attack delay of protections that move code from a client-side application running on an untrusted device onto a secure server, either permanently or temporarily. With these unavailability metrics, we can estimate the loss of information and the delay this causes for an attacker.

With the three types of metrics and the extensive discussion of their relation to attack steps, to protections, and to fundamental software features in the **public deliverables of WP4**, we have proposed a comprehensive metrics framework for evaluation the strength (**potency, resilience, and stealth)** of the wide range of software protections studied and integrated in this project. For several of the proposed metrics, tool support to compute them was developed and integrated into the ASPIRE Compiler Tool Chain. Moreover, tool support was implemented to measure the **cost** (i.e., overhead) of the protections.

The metrics thus cover the four axes of the long established software protection evaluation taxonomy developed by Collberg et al.: potency, resilience, stealth, and cost.

The metrics framework together with the models and evaluation methodology have been presented in two keynote talks and a presentation at the 2014 ARO Workshop on Continuously Upgradeable Software Security and Protection by the project coordinator.

The project has hence achieved all of its goals with respect to the metrics framework for evaluating software protections and for making them measurable.

1.3.4.3 Task T4.3 Experiments with academic subjects

Experiments with human participants are a way to practically observe what we conjectured with the (theoretical) modelling of protections and attacks.

In Task 4.3 we run experiments in “controlled” environments, i.e., in-vitro settings where the experimenter can control (i.e., set) or measure all the relevant variables. This allowed us to elaborate measurable and objective relations between the independent variables (e.g., presence of obfuscation, background on participants) and dependent variables such as success rates and attack times.

We crafted an experimental design that fitted our settings, i.e., doable by (skilled) students in a class time slot. For some student, this required to make them work on source code, so source-level protections have been considered: data obfuscation and client-server code splitting. Both these protections were available when the experiments started.

Several experiments were then replicated in different conditions, to study also variants of the same protections. The different replications included

- Different participants with different background;
- Different variants of data obfuscation (static and dynamic residue number coding (RNC), variable merging, and array reord7ering) and combinations of them (RNC + Array Reordering);
- Different representations of code to be attacked (source code and binary code);
- Different configurations of client-server code splitting (split-small, split-medium).

All four the academic partners conducted two rounds of experiments, and all rounds consisted of two lab sessions, with the following number of participants:

- First round:
    - 16 participants, all Master students, from University of Trento;
    - 12 participants (6 Bachelor/Master students and 6 PhD students/post-docs) from University of East London;
    - 15 participants (14 Master students and 1 PhD student) from Politecnico di Torino;
    - 10 participants (5 Bachelor/Master students and 5 PhD students/post-docs) from Universiteit Gent.
- Second round:
    - 34 participants (32 Master and 2 Bachelor students) from Universiteit Gent;
    - 10 participants (1 bachelor student, 4 master students, 2 PhD students and 3 post-doc researches) from UEL;
    - 19 participants (17 master students and 2 PhD students) from University of Trento;
    - 86 Master students from Politecnico di Torino.

From these experiments, two forms of results were obtained:

- Objective results in terms of success rate and attack time
- More subjective results from open questions about:
    - How participants cope with attack tasks and what attack strategy they adopt;
    - What are the winning attack strategies lead to success and what strategies are not effective;
    - What attack tools are used most;
    - Commonalities between experiments on source code and on binary code (this validate the use of source code to assess protections on code that will be eventually delivered as binary);

The precise qualitative and quantitative results are all documented all in public deliverables, i.e., in D4.03, D4.04, and D4.06.

Moreover, the knowledge obtained from these experiments has been used to tune the data in the ASPIRE Knowledge base, and the results have been included int he validation performed in WP1.

Finally, the experiments form the two publications:

- Alessio Viticchié, Leonardo Regano, Marco Torchiano, Cataldo Basile, Mariano Ceccato, Paolo Tonella, Roberto Tiella. Assessment of Source Code Obfuscation Techniques. In Proceedings of the 16th IEEE International Working Conference on Source Code Analysis and Manipulation, pages 11-20, New York, NY, USA, 2016. IEEE Computer Society.
- Mariano Ceccato. On the need for more human studies to assess software protection. In ARO Workshop on Continuously Upgradeable Software Security and Protection, pages 55-56, November 2014.

With respect to the academic security evaluation experiments, the project clearly reached all of its goals.

1.3.4.4 Task T4.4 Experiments with industrial tiger teams

While the original ideas for this task were very similar to those of the previous task, i.e., to run two rounds of smaller experiments but then do that with professional penetration testers, the consortium adopted a more suitable approach. The main goal of this adaptation was to avoid some of the observed pitfalls of penetration tests conducted on small software pieces protected with few protections. Most importantly, it was decided that such penetration tests, in particular by professional tiger teams, would quickly converge on manual attack paths of least resistance that are actually not very relevant for real-world uses cases, because manual attacks do not scale up to the size and complexity of real-world use cases where many protections are combined.

Instead, three larger experiments, one for each industrial partner, were prepared that consisted of multiple phases. In each phase, the project uses cases protected with different combinations of many protections, were attacked. In two of the three experiments, both internal and external professional pen testers conducted the experiments, which lasted multiple weeks.

The experiments were meticulously designed, prepared, executed, and evaluated, as reported in deliverables D4.04 (June 2016) and D4.06 (end of the project). During the execution of the pen-tests, weekly and bi-weekly conference calls were organized between the designers of the experiments, the project coordinator, and the involved industrial partners and their pen-testers to keep track of the pen-testers' progress and to adapt the plan where necessary. In all of the experiments, multiple attack vectors and multiple combinations of protections were considered.

The pen-testers provided (confidential) reports, and exit interviews were conducted by the task leader (FBK) and the project coordinator based on those reports. The results that can be published (without hurting the commercial interests of the industrial project partners) are all presented in the public deliverable D4.06.

The conclusions of the experiments and the lessons learned were used as a primary input to the validation effort conducted in WP1. Overall, the conclusions confirmed the expert knowledge already available within the consortium regarding the protections' weaknesses and strengths. Most importantly, the protections proved successful in delaying attacks, either by forcing the attackers to perform more manual attack steps, or by pushing them off the original attack paths of least resistance (i.e., the easiest attack vectors on vanilla applications). The results of these experiments have also been used to tune the knowledge in the ASPIRE Knowledge base that was developed in Task T4.1.

Even though a different design was adopted for the professional penetration tests, the original goals of these tiger experiments were achieved.

Even though the project has formally ended, the academic partners are still building on these experiments and the reports. Currently, a joint effort is ongoing to formalize attack models on the basis of the informal reports by means of grounded theory study.

1.3.4.5 Task T4.5 Public challenge

In July 2016, the design of the public ASPIRE challenge was presented in deliverable D4.05. It consisted of eight smaller (diversified) binaries, each of which checked the validity of an input string in a different manner, with the checking code and the checked string being protected with different combinations of protections. The goal set forward for attacks was to reveal the value of the valid input string. This design of the challenges differed from the design proposed in the DoW (i.e., a game anti-cheat engine) because the personnel with the necessary expertise in its domain had left UGent before the public challenge effort was about to start.

A special website and server infrastructure was set up to enable hackers to download challenges, to track those downloads, and to allow the hackers to submit responses (i.e., candidates for the valid strings) and get feedback on their correctness.

In order to attract more hackers, a bounty of 200 euro was put forward for the first successful attack on each of the eights binaries. As not all versions were attacked successfully by the original end date of the challenge (31 Sep 2016), it was extended until the end of the project.

In the end, one hacker captured the bounty for five out of eight challenges. Extensive exit-interviews by email revealed the used attack methods, and mostly confirmed the existing suspicions from the project consortium w.r.t. the weaknesses of the deployed protections. The exit-interview information, as well as insights obtained from all the attack scripts and descriptions that the hacker handed over to the consortium, were used as a primary input to the validation effort conducted in WP1. The results and insights were also presented in the public deliverable D4.06.

With the launch, execution, and result analysis of the public challenge, we reached all goals set forward in the DoW for this external security evaluation validation task.

1.3.5 WP5 Framework (M04-M36)

The designs of the ASPIRE Framework and the evolution over time have been extensively documented in deliverables D5.01, D5.03, D5.06, D5.07, D5.11, and D5.13. The software prototypes, evolving from the first basic implementation to the complete implementation via continuous integration, were delivered in D5.02, D5.04, D5.05, D5.08, D5.09, D5.10, and eventually open sourced as D5.12.

1.3.5.1 Task T5.1 ASPIRE tool chain

To protect applications with the ASPIRE protections, we designed and implemented a tool flow that automatically transforms annotated C code into a protected binary application or library. This tool flow applies the different source-level and binary-level protection techniques in the correct order. To that extent, the tool support for the different protections function as plug-ins.

Additionally, this tool flow includes computing metrics associated with the protected application, and correctly deploying the applied protections on protection servers in the case of server-assisted, inline protection techniques. This tool flow is driven by a central component called the ASPIRE Compiler Tool Chain (ACTC), which is programmed in Python on top of the open-source DoIt task automation Framework.

We designed and documented a set of source code annotations with which a tool chain user can specify the protections to be deployed, the fragments on which they should be deployed, and the parameters to be used. The tool flow extracts these annotations to steer all the protections.

The tool flow consists of twenty different steps in total. All these steps have different input and output files, all of which have been extensively documented. In the source-level part, all plug-ins actually produce either plain C code or pre-processed C code. So in essence all source-level plug-ins are source-to-source rewriters. In the source-level processing and in the binary-level processing, which is mostly based on link-time rewriting tools developed on top of UGent's link-time rewriting framework Diablo, the ACTC correctly propagates all the output files produced by the enabled steps to the correct input arguments of the following enabled steps. Passes can be skipped if the necessary plug-ins are not available or if specific protections need not be applied.

The ACTC implementation is highly configurable by end users, and can be instructed to enable/disable the individual steps and to configure their invocation by means of human-readable JSON files. This use of JSON files for configuration forms a user-friendly alternative to a GUI, which would have required much more effort to implement.

All of the discussed design aspects facilitate the addition of additional protections and plug-ins in the future, or replacements of existing ones.

To improve the performance of the tool flow when it is used to evaluate different combinations of protections (as can be requested by the ASPIRE Decision Support System), we also designed and implemented a caching system. This caching system keeps track of the annotations that each plug-in consumes, thus enabling it to reuse a plug-in's cached output if nothing has changed to the its input code and its annotations, rather than re-invoking the potentially time-consuming plug-in.

The source-level part of the tool flow currently implements the following protection steps in the listed order: insertion of annotations generated by the ADSS, white-box cryptography, source code pre-processing, data-hiding obfuscations, client-server code splitting, source-level transformations required for offline code guards, anti-cloning protection, reaction unit insertion, (SLP10), diversified cryptography library insertion, control flow tagging, source code annotation extraction for the binary tools, and generation of the source parts of remote attestation.

Next in the tool flow, the final set of source files is compiled and linked with a slightly patched standard compiler (gcc or LLVM). Depending on which protections have been enabled and applied, additional source files are compiled and additional libraries are linked in.

The binary object files are then further analysed and protected with different runs of the Diablo link-time rewriter. As the ASPIRE project targets ARMv7/Android binaries, UGent extended the support of their Diablo tool to include ARMv7 instructions and to add support for Android. Furthermore, UGent wrote several small patches for the already mentioned modern C(++) compilers and for the GNU binutils such that the generated binaries can be transformed correctly with Diablo.

During the different runs of Diablo and other invoked binary-level tools, the following protection deployments and processing steps are executed: a self-profiling version of the application without binary-level protections applied is generated and run on a development board, extractable chunks for the SoftVM protection are identified and extracted, the X-Translator creates bytecode and stubs to replace the extractable chunks. These bytecode and stubs are then linked with the application, which is then protected in a final run of Diablo. Diablo then applies the following binary protection steps in the listed order: SoftVM bytecode integration, call stack checks, binary obfuscations, code mobility, code guards, and remote attestation.

Furthermore, different metrics are computed. Static metrics are automatically generated by the different binary-level steps. However, to compute dynamic metrics, the protected application needs to be run on a development board, and it needs to be instrumented in such a way that profile information is collected. We extended Diablo to generate self-profiling versions of protected binaries, and the ACTC includes additional steps to automatically run the self-profiling binaries on development boards, and to collect and process the produced profile information.

Some of the protection techniques have a server-side component. The ACTC invokes scripts to update the relevant databases and directories with the information associated with the protected binary. The following server-side scripts are invoked in the listed order: code-splitting deployment, code mobility deployment, and remote attestation deployment. Finally, the ACTC has support for renewing some of the protections as a means to implement diversity in time. It generates scripts to renew the white-box cryptography tables, and scripts to renew the mobile code blocks through diversifying the binary code of the blocks.

Also worth mentioning is that the ACTC has extensive logging support: all source-level and binary-level transformations are logged, and for the binary-level transformations, control flow graphs of the code fragments before and after each transformation step can be generated on disk.

The ACTC and the integrated protections were validated successfully on the project use cases, as discussed in Section 0. All techniques developed in WP2 and WP3 and foreseen to be integrated into the ACTC in the DoW have been integrated, and more (some forms of RA were not foreseen to be integrated). As discussed in Sections 0 and 0, excellent composability of the integrated protections has been achieved in the ACTC's plug-in architecture.

We can hence conclude that the project achieved all of its protection tool chain integration goals.

To achieve that level of integration of contributions from all project partners, a major effort was invested in the development of a common build environment consisting of a VM image and many scripts to update and maintain the image consistently at all partner sites. This VM facilitates the continued collaboration between project partners after the end of the project, and it also formed the basis of the Docker container that was developed for the open-sourcing of ASPIRE software as will be discussed in Section 0.

1.3.5.2 Task T5.2 ASPIRE Decision Support System

The ASPIRE Decision Support System (ADSS) is a framework whose aim is to help software protection experts in automating the process of protecting software applications. Starting from the annotated source code, the ADSS generates the golden combinations, i.e., the set of protections that best mitigate the risks against the application assets, and drives, by means of protection specific annotations, the ACTC to apply the golden combination. The ADSS also outputs logs and reports that explain the entire decision process.

The golden combination is selected with a complex work-flow. Initially, all the information about the application to protect is gathered using compiler tools from the source code. A series of analyses tools extract significant information form the application to protect by means of static analysis tools (e.g., CodeSurfer or CDT). The parts of the application that may deserve to be protected are identified, listed and related with the ASPIRE Security Model with an instantiation of the ASPIRE Knowledge Base which completes the knowledge about the application to protect for risk analysis purposes.

Then, the threats against the application assets are determined via a backward reasoning system that identifies all the attack paths. The reasoning system can be fine-tuned by enabling analysis based on different types of attackers (based on their expertise, available tools) and attack strategies (static vs. dynamic, compromising of the attacker or victims' copy), and by limiting the search space to have results in useful time.

Next, the possible mitigations to the identified attack paths (i.e., the software protections that may delay attacks against the target assets) are identified by means of further inference methods. However, protections in isolation are not the most effective way to protect applications' assets, as protection show synergies (which have been carefully analyzed during the project) that, once correctly exploited, strengthen each other.

The golden combination is selected by means of an ADSS component that implements a game theoretic approach based on customized minimax search tree (named level 1 protection). Several pruning and reduction techniques have been supported in order to reduce the search space and find golden combinations reasonably close to the real optimum in useful time (alpha-beta pruning with aspiration windows, iterative deepening with transposition tables, razoring, futility margin, extended futility margin and reductions based on the node scores). Evaluation of protections effectiveness (in isolation and in combination) used by the optimization function has been built on expert judgments formally captured in the knowledge base. Judgements are tightly coupled the metrics computed with the ASPIRE metrics framework which are used by a fast (linear) evaluation engine that has been implemented to estimate the effectiveness of combinations of protections to meet the very strict requirements of the ADSS (i.e., to evaluate millions of combinations of protections in seconds). The engine accuracy is configurable to meet the users' requirements and a proper performance vs. precision trade-off.

Finally, an optional ILP optimization model (named level 2 protection) finds the best way to hide the protected assets and delay attacks. Assets are hidden by protecting with same protection techniques in the golden combination other parts of the application or by extending the protections proposed in the golden combinations to areas outside the assets, until a set of user defined overhead constraints are saturated.

The ADSS has been trained with the inputs and feedback from the experts in the ASPIRE project, in order to allow the tool to first determine reasonable combinations of protections then to fine tune the entire optimization process towards optimal protection. Then, during its extensive validation, the interaction with several experts has further improved the ADSS ability to propose excellent protection.

The ADSS has been open-sourced at https://github.com/SPDSS/adss. The ADSS foundations have also been published and presented 10th IFIP WG 11.2 International Conference.

With respect to the challenging goal of developing a decision support system to help a protection tool chain user obtain measurable protection at a controlled level of overhead, the project has achieved its goal. While the ADSS might not be ready yet for commercial use, it has shown great potential, and with its availability, the project delivered a great research tool for further improving the state of the art with respect to software protection decision support.

1.3.5.3 Task T5.3 Open Source

Near the end of the project, the coordinator's team has set up an ASPIRE-FP7 **GitHub repository** at https://github.com/aspire-fp7 that makes the **vast majority of the prototype software** developed in the project and integrated into the ACTC available under open-source software licenses. In particular, all protections contributed by academic partners and integrated in the **ASPIRE Compiler Tool Chain** are available, with the exceptions of protections being exploited in a spin-off from FBK. Also the **ASPIRE Decision Support Systems** (full and light) are available through the repository.

Some components are released in the main aspire-fp7 repository, other components are released in other repositories and linked by the main repository. The reason is to allow for accounting of the downloads in the future, and for facilitating future development of components outside the context of ASPIRE.

From the repository, third parties can retrieve all components necessary to deploy the ASPIRE Compiler Tool Chain on their software. Moreover, a **Docker container** setup has been developed and released to facilitate setting up build and development environments in which to deploy the ASPIRE Compiler Tool Chain.

Public deliverable D5.13 documents the repository structure and the Docker container, and describes how to setup and deploy the tools on a small example. This document hence serves as a **manual** to the open sourced ASPIRE framework, together with the public deliverables of the project, in particular D5.11 which provides a comprehensive overview of all aspects of the ASPIRE Compiler Tool Chain and the ASPIRE Decision Support System, and D1.04, the ASPIRE Reference architecture.

With the GitHub repositories of the open sourced prototypes, the Docker container to ease the take-up, and the public deliverables serving as manuals, the project has achieved all open-sourcing goals.

1.3.6 WP6 Use Cases - Demonstration (M04-M36)

In three tasks, the industrial partners developed and implemented the three use cases they had designed in Task T1.1 at the beginning of the project.

In all use cases, the components to be protected with ASPIRE protections are native dynamically linked libraries. In the Digital Rights Management (DRM) use case contributed by NAGRA, two such libraries are loaded as plug-ins into the Android DRM and media server processes. There, the perform the necessary video decryption and DRM license management necessary to play videos in a video player that NAGRA developed for this project, along with the two libraries themselves.

In the One-Time Password (OTP) use case, contributed by GTO, the purpose is to generate a temporary password called OTP that can be used to authenticate later on a banking site or any business site that needs a stricter authentication process than the static identifier/password method. The application is split in two phases. During the provisioning phase a secure channel is opened with the OTP server to retrieve various data including the Device Key that is used in the second phase to generate the OTP values. The provisioning phase can be executed only once while the OTP generation in the second phase is required each time the user wants to authenticate on the web site. As a real mobile application, there are some constraints: user interactions are managed by the GUI that is specific to Android; and the application needs to access a Service Provider server. In this use case, the security sensitive functionality is implemented in the library that needs to be protected with ASPIRE protections, and this library is embedded in an Android Application Package, together with the GUI frontend. GTO developed that app, the library, and the necessary server support for the provisioning phase.

In the Software License Management use case, contributed by SFNT, a license manager is implemented in a dynamically linked library. This library is linked/loaded into software applications to manage the credentials and access to restricted content or functionality. In the toy example application developed by SFNT, the content are secret answers to quiz questions. In one developed form, the application consists of a GUI Android Dalvik app, and the library to be protected is embedded in the app's package. In another developed form, the app is a Linux command-line application, for which the library is loaded on demand by the Linux. SFNT developed both versions as well as the library itself. Having versions for two platforms helped the project in ensuring that the developed technology can be exploited on multiple platforms.

All industrial partners adapted their specific build systems to enable the use cases' compilation and protection with the ASPIRE Compiler Tool Chain. The industrial partners identified the assets in the use case code, and together with the academic partners, source code annotations were developed to let the tool chain deploy a combination of protections as an expert would choose manually. Where necessary, minor modifications were made to enable the deployment of the ASPIRE tool chain and its plug-ins.

Protected use cases (with combinations of many protections) were prepared for the tiger experiments of Task T4.4, for the validation in Task T1.5, and for demonstration.

In the end, al foreseen and developed software protection technology can be deployed, evaluated, and demonstrated successfully on the three use cases. The project has hence achieved all of its goals with respect to the use cases.

In year 3, the consortium drafted and implemented a demonstration plan. The consortium took up the recommendation from the project reviewers to create some videos to disseminate the project results to industrial target audiences, and expanded this idea to cover all demonstrable results of the ASPIRE project, plus some overall presentations on the project goals, its outcomes, lessons learned, etc.

In total, 30 videos will eventually be created, of which 29 will be released on the ASPIRE-FP7 Software Protection Demonstration video channel at https://www.youtube.com/channel/UCntMGBjHr_oW5wEd5JgjD6g. In total, those public videos will provide **over three hours of demonstration material**. At the time of writing of this report, 22 videos were already published, totalling more than two and a half hours. The rest will be published before the final project review. In addition, all partners can of course still use the project use cases to provide live demonstrations to selected audiences.

With the many demonstration videos on the YouTube demonstration channel, and the available live demonstration options, the project has achieved all its demonstration goals.

1.4 Potential Impact, Dissemination Activities and Exploitation of Results

1.4.1 Main Impact

1.4.1.1 Impacts expected in the work programme

Of all the users and stakeholders that have assets stored on mobile devices with which mobile services are accessed, **ASPIRE focuses on protecting the assets of the content and service providers.** The ASPIRE results contribute to the expected impacts from the ICT 2013.1.5 Trustworthy ICT objective, which we quote and then discuss here in detail.

> *"Demonstration of secure and privacy-preserving technical solutions in clouds, mobile services and management of cyber incidents applying state-of-the-art research results, ..."*

State-of-the-art results: A range of state-of-the-art technical solutions covering five lines of defence has been developed to mitigate a wide range of attacks on the assets and software of mobile content and service providers. The result of this R&D have been integrated into the ASPIRE Framework consisting of the ASPIRE Compiler Tool Chain and the ASPIRE Decision Support System.

Mobile services: ASPIRE demonstrated the framework and its security solutions on three real-world use cases from the industrial partners' domains in mobile services. The demonstration is not limited to the concepts of the proposed solutions, but to the fully functional tool chain and its decision support system.

Privacy-preserving technical solutions: When a mobile app is designed correctly to protect the privacy of data, then ASPIRE's anti-tampering protections, which are themselves reinforced by the other protection layers, are able to prevent modifications that subvert the intended privacy-preserving behaviour. Moreover, ASPIRE results make it much easier to develop such correctly designed apps, as the ASPIRE tool chain facilitates and automates the use of data hiding techniques such as white-box cryptography and advanced data obfuscation.

Management of cyber incidents: ASPIRE's remote attestation solutions enables server-side monitoring of the clients connected to a service. A provider can keep track of the trustworthiness of the client's software execution and manage incidents. This can be done semi-automatic, with policies that enable/disable services to clients where malicious activity is detected. ASPIRE's renewability also allows automatic upgrades to be integrated in those policies.

" ... ensuring interoperability ..."

Interoperability of services and access devices: By raising the level of protection against man-at-the-end attacks, ASPIRE results help to secure what is currently considered the weakest point in many data networks: the (mobile) end devices. ASPIRE improves interoperability of devices by enabling the provision of security services that previously required custom secure hardware components to meet the security requirements on a zoo of untrustworthy mobile devices and platforms.

Interoperability of servers and client devices: The technology developed in ASPIRE to integrate and set-up the server-side logic of many online protections, and the common communication layer that was developed constitutes an important abstraction that facilitates interoperability.

Interoperability of networks: ASPIRE's network-agnostic protections are deployed 'over the top'. They do not impose any requirements on the network beyond additional bandwidth that can be controlled and adapted.

Interoperability of protections: As has extensively been documented in Section 3 of this final project report, the ASPIRE Compiler Tool Chain integrates tool support for many protection, spanning the foreseen five layers of defense. And as discussed on multiple occassions, the supported protections are highly composable. Clearly, the interoperability of software protections has been given a boost by the ASPIRE results.

"... and compliance with privacy legislation."

Privacy against competitors: By providing mobile service providers with a framework to protect the privacy of their assets installed on mobile devices, ASPIRE results help those providers to ensure compliance with privacy legislation. It will help them, e.g., by letting the software-based protection ensure that the legal provisions in end-user agreements regarding illegal analysis by competitors, unfair and illegal use and leakage by end-users, illegal modification and redistribution by cybercriminals, and against combinations thereof, are complied with.

Legal compliance: ASPIRE's protection strength evaluation methodology and tool support, while still requiring more research and development, will eventually enable service and content providers to operate in an environment dominated by legally binding security requirements.

"Widen take-up of research outcomes by service providers and wider adoption of ICT security solutions by European companies and users."

Take-up by partners: The industrial ASPIRE partners will use the project results to expand their own product portfolios and the portfolios of their customers to face the competition and become more independent from big DRM companies like Google, Apple and Microsoft.

Take-up by SMEs and increasing the number of European spin offs in the ICT field: The automated, integrated, and open-sourced ASPIRE tool chain can help in lowering the market entry ticket price and the time-to-market for SMEs and spin offs by freeing them from investing effort and time in protection, and enable them to sell protected software without facing diversification and deployment costs related to hardware-based solutions. The portability of the ASPIRE tool chain enables a unified software protection strategy, lowering the cost to secure software on a plurality of devices. Furthermore, the ASPIRE tools can open up new business opportunities such as pirate activity monitoring based on the remote attestation activities; or dynamic response and patching based on renewability mechanisms.

Take-up by service providers: Once matured, the ASPIRE Decision Support System can enable software providers, their content providers and ISVs —all of which can use ASPIRE technology— to make an informed selection for the protection level of their applications. ICT market leaders adopting software-only protections with demonstrated qualities will give a strong signal to increase the confidence in such solutions and will help a wider adoption. ASPIRE's enabling of much cheaper any-device-any-where secure access functionality will enable content providers to protect their content very strongly on a wide range of devices. They therefore will no longer risk their customers becoming annoyed because by fair use restrictions. As a result the customers will become aware that running legitimate, protected, reasonably priced software is preferable over living at risk with an uncontrolled environment on their device, which in turn leads to an improved customer relationship to the benefit of the service providers.

Take-up in traditional sectors: Companies in sectors such as banking risk having to refund customers that became the victim of security breaches. The banks' responsibilities are so high that they cannot take the risk of validating the trustworthiness of their customers' computers (i.e., hardware and complete software stacks) to perform and authorize transactions. So instead those banks have to really on expensive, cumbersome hardware devices such as digipasses to authorize online banking transactions. ASPIRE's remote attestation technology can help in ensuring the credibility of applications by proving their authenticity irrespectively of the trustworthiness of the underlying platform. This may enable sectors like banking to migrate to pure software-based online services.

> *"Unlock the market restrictions, reveal the incentives to create a functioning cyber security market and increase the number of European spin offs in the field."*

Unlock the market restrictions: Europe faces the major problem that software needs to be secured on mobile platforms that do not give user-space applications access to hardware security. For example, most mobile devices include secure boot mechanisms but do not let third-party vendors benefit from it. Such market restrictions can only be mitigated with pure software protection techniques, as provided by the ASPIRE project.

Incentives for alternative secure markets: Validated metrics and security models, together with provably strong white-box encryption as developed in ASPIRE will increase confidence, turning protection against man-at-the-end attacks into a science long after cryptography started protecting against man-in-the-middle attacks. The real-world potential of software protection as an important aspect of trustworthy ICT has become measurable through the ASPIRE metrics and security evaluation tools, thus revealing the true potential of software-based protection as an incentive for further development of a functioning software-based cyber security market.

Incentives for alternative inexpensive markets: ASPIRE's incorporation of renewability as a core protection concept ensures that successful attacks (if any) can be stopped from doing damage extremely fast, which will lower the cost of successful attacks considerably. This reveals another important incentive, as does the significant cost and effort reduction that ASPIRE solutions will bring for providing adequate protection. As such, access to the ASPIRE tool chain can boost the application development economy.

Incentives for market players: Besides revealing the aforementioned incentives to create a market, ASPIRE's solutions will also allow individuals and companies, i.e., the market players, with potentially good software protection ideas to evaluate the true potential of their ideas. They inherit all the ASPIRE framework to work with; its metrics and measurability, highlighting the weakest points, can permit the players to select the best areas to invest. The open source availability of all evaluation software developed in the project certainly helps here.

> *"Development and implementation of European strategies for internet security."*

Security as a practice: Whereas software protection in practice still suffers from an almost complete lack of best practices and generally accepted evaluation and testing methods, the

ASPIRE protection evaluation and modelling aims for realizing a de facto standard for software protection models and evaluation practices. The first steps in that direction have now been taken.

"Significant contribution to making Internet a medium that can be used to exercise human rights, including in hostile environments. "

Right to share information: The ASPIRE results can help with the development of safe-to-use social media applications and services that can run on any device anywhere and that play a vital role to document violations of human rights and to organize resistance activities in hostile environments where human rights such as the freedom of opinion and expression (Universal Declaration of Human Rights, Art 19) and freedom of peaceful assembly (Art 20) are at stake. The combination of ASPIRE's white-box crypto with renewability will enable applications such as image capturing/sharing, planners and social media to overcome unacceptable restrictions to those freedoms. ASPIRE's white-box cryptography and anti-tampering technology, protected by the other layers of defence, will also enable the distribution of videos captured by trusted applications, i.e., videos that are guaranteed to be free of editing and of which the embedded recording place and time (obtained through, e.g., the GPS) cannot be disputed.

1.4.1.2 Macro-Economic and Societal Impact

Cyber security is a growing economic and societal concern, as is also witnessed in the recent initiatives such as the EC cyber security PPP and the creation of the European Cyber Security Organization (ECSO).

Since the start of the ASPIRE project, the use of mobile devices has further exploded. Not only do many people use tablets and smartphones on a daily basis, also connected camera's and smart watches are omnipresent these days.

The market for mobile software, ranging from simple apps to full blown online applications, is therefore still growing fast. And so is hence the market for software monetization technologies, such as license management technologies.

Also in the entertainment industry, traditional distribution and consumption patterns are gradually but quickly being replaced by online methods. The rise of Netflix and the availability of the programmes of television networks via online streaming are just two examples thereof.

Online banking has also grown significantly, with not only traditional banking services such as money transfers being handled in online apps these days, but also with innovations such as apps that help people to automatically split bills.

In all of the mentioned technologies, security-sensitive assets are embedded in and handled by the software. A growing fraction of the European economy (of service providers, content providers, and software providers) hence has become a stakeholder in mobile software, and in the protection of its assets in that software. And so have the consumers and end users of the mobile software.

In the future, the importance of secure mobile software will only continue to grow and expand to more and more sectors. For example, the insurance industry is studying how to exploit the capabilities of mobile devices that track their customer's behavior (e.g., in traffic) and health. And as mobile computing devices take over manual tasks of safety-critical systems (e.g., autonomous vehicles), the trustworthiness and hence the attestation of the control software will become critical to handle disputes following accidents.

It is clear that in the short-term future, the success or failure of the protection of assets embedded in mobile software will impact all European citizens, and many sectors. The latter is also clear of the list of sectors assembled in Section 2.2.3 that can benefit from protected assets, be it in mobile or in other software.

Even if only a fraction of ASPIRE's improvements of the state of the art reach the market, its improvements in the strength of software protection techniques, its improvements in making this strength measurable, and its improvements in easing the deployment of combinations of software protections through its (largely open-sourced) framework, will have a huge economic and societal impact.

In terms of euros and the concrete potential for economic growth, we can quote the European Cybersecurity Industry Proposal for a contractual Public Private Partnership report produced in June 2016 by the newly founded European Cyber Security Organisation: "Cybersecurity is one of the fastest growing ICT sub-domains as the critical processes in the society are becoming increasingly dependent on IT solutions. The sustainability of most important government processes, provision of vital services and even day-to-day actions of all the citizens rely more and more on well-working and secure ICT solutions. [...] The main difficulty is the evaluation of the percentage of the world market secured by companies having their origin in Europe (not being European subsidiaries or European HQ of external companies). A rough estimation, gives a value of about 6 bln €, corresponding only to 35% of the European market and 8,5% of the world market (a value close to an estimation given by Gartner specific to “system and network security software” sector), thus showing the progress that the European cybersecurity industry can make! The corresponding number of highly skilled experts in European cybersecurity industry is suggesting a figure of about 100.000 direct jobs.". The report also states "According to the market study made in the F7 project IPACSO the European civil cybersecurity market reached the level of €18.8 bln in 2014 (yet, including Russia). It is forecasted to grow at an annual rate of 7.4% reaching the level of €26.7 bln (including Russia) in 2019. The European market is however smaller than the US market which is estimated to be around €26 bln in 2014. The UK, Germany and France constitute the biggest market sub segments and about 60% of the European market and estimated to achieve growth rates of between 5-6%." There is hence a clear potential for economic growth in the software protection sector and all sectors that consume its products.

Obviously ASPIRE has not solved all problems, and more research and development is needed to arrive at satisfactory, complete solutions, but the ASPIRE consortium is confident that the project has contributed significantly towards the realisation of such solutions, and that the ASPIRE results will be taken up in future research projects and products. With its contributions, this project provided a direct response to one of the major societal challenges of our time.

1.4.1.3 Business Impact on Participating Industrial Partners

The impact section of the DoW of the ASPIRE project already projected market sizes and shares, as well as trends for the economic sectors in which the industrial ASPIRE partners wish to exploit the results of the project, i.e., in which they intend to gain a competitive advantage by exploiting the project results.

The market sizes projected in the DoW need to be revised significantly. For example, the over-the-top and multiscreen media market sizes were estimated to be around EUR 0.5 billion in 2017 in the DoW, but are estimated to be more than 100 times larger in 2017 in recent market analysis reports.

Detailed market analyses are reported in the confidential deliverable D7.06 Exploitation Report, but the precise numbers and the specific markets in which the industrial partners plan to exploit the ASPIRE results cannot be disclosed publicly here.

Still, it is clear that competitive advantages, or even simply keeping up with the international competition, will generate millions of euros of income for the industrial ASPIRE partners.

1.4.2 Dissemination Activities

The DoW concentrated on three aspects: raising public awareness, dissemination of results, and exploration preparation. These aspects covered different methods and activities that needed to be initiated in order to achieve the goal of establishing ASPIRE as a successful and sustainable project.

Later sections in this report discuss and list the different dissemination activities of the project. Public deliverable D7.05 Dissemination Report presents them in even more detail. The executive/highlights summary is as follows:

- a project logo;
- Word and LaTeX document templates and a PowerPoint presentation template;
- a 4-page A4 project leaflet and an A0 project poster;
- the project website;
- an open source repository;
- 25 scientific publications (of which 17 peer-reviewed);
- 18 participation activities in Conference or Workshops, incl. keynotes and an ASPIRE tutorial;
- 35 presentation activities where ASPIRE results were disseminated to expert audiences;
- 29 demonstration movies on YouTube;
- the organization of 2 international workshops on software protection, co-located with top conferences (ICSE 2015 and CCS 2016);
- 11 dissemination activities to the general public, including press releases taken up by ACM TechNews, and an interview with the coordinator broadcasted on Flemish local public television at the time of the project kick-off meeting.

1.4.3 Exploitation of Results

1.4.3.1 Exploitation activities in the ASPIRE project

Section 5 in the confidential Deliverable D7.06 lists the exploitation activities already performed by each partner during the project. This in line with the observation that the main exploitation of ASPIRE results was and still is foreseen through each partner's own organization. For the industrial partners, a more extensive description of their confidential plans and actions already taken can be found in the annexes of D7.06. As the companies are competitors, their annexes are not accessible to one another.

A consortium-wide activity already conducted to foster exploitation of Foreground by the industrial partners as soon as the project is finished, is the SWOT analysis and TRL assessment of all developed technology in D7.06. Moreover, to maximize the exploitation potential in de academic and industrial research communities, the project invested heavily in open-sourcing its software prototypes, in providing documentation for the open-sourced software, and in producing and publishing 29 demonstration videos on YouTube.

Other exploitation highlights from within the project duration are

- FBK's plans and ongoing activities towards the creation of a start-up centred around their data obfuscation and client-server code splitting Foreground;
- UGent's transfer of obfuscation technology to Samsung Research UK, for which the project coordinator was awarded a HiPEAC Technology Transfer Award;
- UGent's ongoing discussions with a company to transfer its anti-debugging IP developed in the project;
- POLITO's agreement with Christian Collberg (University of Arizona) to integrate the remote attestation developed during the ASPIRE project into the Haathi framework (http://haathi.cs.arizona.edu/).

1.4.3.2 Contribution to standards

Throughout its development and integration activities, the consortium has put high priority on the use of existing tools, processes and formal and de facto standards:

- The ACTC can handle programs written in the standard programming language C.
- The code generated by the ACTC source-level protection tools will meet the C standards.
- For the ACTC Tool chain, users can select multiple compilers. Tests have been conducted with several generations of LLVM, GCC for both Linux and Android targets.
- The native binary code formats used by the ACTC are standard object file formats.

- The means used to pass information about code fragments from one tool to another rely on standard forms of information, such as relocation, symbol, and debugging information as defined in the object file standards.
- The configuration files used to configure the tool chain are standard XML and JSON files.
- The generated programs only rely on commonly available features in standard libraries and operating systems. This was tested for, e.g., Android and the anti-debugging techniques and the code mobility techniques.
- Compatibility of the developed protections with the latest releases of those OSes was also tested.
- To facilitate the take-up of the open-sourced software, industry standards such as Docker containers are used.

No formal activities targeting standardization organizations and bodies have been initiated yet in the ASPIRE project.

1.5 Miscellaneous

1.5.1 Project website

To serve the broadest possible visibility of the project, the project website was launched in the first month of the project. All pages on this public website are available to everyone, it is not necessary to login.

The public part of the website consists of the following pages:

- **Home**: General introduction to the project, brief overview of consortium by means of partner logos.
- **Consortium**: Description of all project partners and principal investigators
- **Contact**: Contact form
- **Resources:** container of all public resources:
    - **ASPIRE papers**: includes published papers, all public project deliverables which have been accepted and other resource, such as videos;
    - **Knowledge base:** includes published papers as well and related sites interesting for the project.
    - **Project Deliverables:** includes a list of ASPIRE project deliverables;
    - **Source code:** contains the link to the GitHub repositories in which most of the ASPIRE code has been published with open-source licenses;
    - **Demo Videos**: contains the link to the ASPIRE YouTube channel with all the videos on different project results;
    - **Other Resources**: contains various resources, such as a TV interview, project leaflets, keynote talks, invited lectures, and press-releases.

The ASPIRE website allows consortium members to register and log in to the website. After doing so, the private part of the website can be accessed.

The private part of the website includes:

- **Wiki** pages used for internal dissemination of relevant information that needs to be updated regularly;
- A **Steering Board** page linking and listing all information regarding the boards' meetings, such as agendas and minutes;
- An **action tracker** page listing all ongoing actions, deadlines, progress states, etc.;
- **mailing list archives**;
- The project **SVN repository for documents**;
- The project **SVN repository for source code.**
- An **EC downloads** page where reviewers and the EC program officer could access all relevant reports.

The ASPIRE project website is available at https://www.aspire-fp7.eu.
The project website was updated continuously by the project Coordinator, whereas all partners participated in the process by notifying the Coordinator of important news, publications and developments. Detailed website statistics have been presented in deliverable *D7.05 Dissemination Report.*

1.5.2 ASPIRE Logo

In order to establish the immediate recognition of the ASPIRE project, the official project logo was designed. The logo was used in all dissemination tools from internal documents and reporting templates to external communication tools such as the website, presentations and brochure.

Along with the logo, two buttons were developed that could be used in all kinds of graphical dissemination material.

1.5.3 ASPIRE Templates

In order to streamline the dissemination of ASPIRE results and create a recognition of ASPIRE graphical material in the software protection community, a Word template was created, and a similar looking LaTeX style was also created along with a PowerPoint presentation template. The templates, which were prepared at the beginning of the project also helped to save time and effort for the members of the consortium, since no further design work was necessary.

1.5.4 ASPIRE Leaflet

As soon as the project had started, the communication company Magelaan was hired to design a project leaflet that can be handed out by the project partners at networking events. This flyer was distributed at multiple local and international events by multiple project partners. A digital copy is available on the project website.

1.5.5 ASPIRE Poster

On the basis of the project leaflet graphics, we also designed (internally) a general ASPIRE poster that could be reused by all partners at poster events. A digital copy is available on the project website.

1.5.6 ASPIRE Social Media

Social media can help in spreading project-related information to a wide audience. They are therefore a valuable tool to disseminate project ideas and results. To start using social media, we waited until enough results were becoming available, to avoid so-called sleeping social media accounts.

In September 2014, the ASPIRE FP7 Twitter account was launched (https://twitter.com/aspirefp7), where project members could tweet about the project and related subjects.

In November 2014, the ASPIRE FP7 LinkedIn group was launched (https://www.linkedin.com/groups/Aspire-FP7-7300827), for stakeholders and other interested people.

On the ASPIRE website direct links to the ASPIRE Twitter Account and the LinkedIn-Group can be found.

1.5.7 Software Protection Workshops

With two successful workshops, the consortium organized one more workshop than foreseen in the DoW.

ASPIRE Workshop: 1st International Workshop on Software PROtection

The first Software PROtection (SPRO) workshop was co-located with the ACM/IEEE International Conference on Software Engineering (ICSE) in Florence (Italy) on 19th May 2015, one of the three top conferences in software engineering.

In this workshop, Paolo Falcarin served as general chair and Brecht Wyseur served as program chair. They first assembled and submitted a proposal for a workshop to the ICSE workshops chairs, and after the workshop proposal was accepted, they assembled the program committee (in which all ASPIRE Principal Investigators were involved), and launched the workshop website at https://aspire-fp7.eu/spro/.

The programme committee of experts followed a very thorough review process to ensure high quality papers and presentations. Each research paper was reviewed by at least four program committee members. We received 19 submissions from 60 authors of 13 countries. The nine best papers were accepted as full papers for publication and presentation at the workshop; of the accepted papers, 12 authors were from industry and 18 from academia.

Five of the nineteen papers submitted came from the ASPIRE consortium, and three of them were accepted. About 40 people registered for the workshop, coming from North America, Europe, and Asia.

We also included in the final workshop programme two keynote talks: one from Prof. Bart Preneel in the morning and one from Prof. Bjorn De Sutter in the afternoon. The first keynote from Prof. Preneel provided an overview of the challenging problems faced by software security, while the second keynote from Prof. De Sutter (the project coordinator) presented the ASPIRE initial results, and introduced the overall aims and objectives of the ASPIRE framework.

ASPIRE Workshop: 2nd International Workshop on Software PROtection

The second SPRO workshop was co-located with the ACM CSS conference in Vienna on 28 October 2016, one of the three tier-1 conferences in the domain of computer security.

For the second workshop, Brecht Wyseur served as general chair and Bjorn De Sutter served as program chair. They first assembled and submitted a proposal for a workshop to the ACM workshop chairs, and after the workshop proposal was accepted, they assembled the 25-person program committee (including ASPIRE Principal Investigators), and launched the workshop website at https://aspire-fp7.eu/spro/.

Fourteen papers were submitted, of which three from within the ASPIRE consortium. Of those, eight papers were accepted, of which one from within the ASPIRE consortium.

About 80 people registered for the workshop, coming from North America, Europe, Asia, and the Middle East. The vast majority of the registered attendants did show up, with the peak attendance being about 55 people.

The ASPIRE tutorial presented by Bjorn De Sutter and Cataldo Basile proved a great way to disseminate the ASPIRE outcomes, and to introduce the ongoing open-source effort and the publication of demonstration videos on the project's YouTube channel.

The workshop was concluded with a very interactive panel discussion. At the end of the panel discussion, the workshop organizers also announced their intention to keep organizing the SPRO workshop in the years to come, and people were invited to join the SPRO steering committee. If possible, the workshop will be aligned with other efforts in the new H2020 EC PPP on Cyber Security and the activities of the new European Cyber Security Organization (ECSO, http://www.ecs-org.eu/). Offers to join forces with PPREW/SSPREW were discussed, but they were considered suboptimal: bringing together European software protection industry experts and academics will only be successful when done within Europe.

The workshop ended with a dinner sponsored by NAGRA, to which all attendants could participate if they registered. In the end, about 30 people attended the dinner, resulting in long and interesting discussions of many topics relevant to the workshop and to the ASPIRE partners and research.

1.5.8 YouTube Channel

In September 2016, the ASPIRE-FP7 Software Protection YouTube Channel was launched (https://www.youtube.com/channel/UCntMGBjHr_oW5wEd5JgjD6g). This channel shows (will show) about 30 video demonstrations of project results.

1.5.9 Open Sourcing

Part of the source code developed in the ASPIRE project has been open sourced. We also created an online presence for the open source parts of the project.

We decided to put all of our source code in git repositories. We created an 'aspire-fp7' team on the GitHub repository sharing website as the central point for our repositories, which can be found at https://github.com/aspire-fp7.

Most of the code written by the academic partners is open sourced. The only exception is the source-to-source techniques developed by FBK, as they are planning to commercialize this technique in a spin-off. The ACTC, which is joint work between NAGRA, GTO, and UGent, has also been open sourced.

We have also written and published scripts to set up and to run Docker containers that contain all of the open sourced tools. Docker is a lightweight Linux virtualization technology. Users can thus clone our docker repository at https://github.com/aspire-fp7/docker, run the scripts, and immediately start running the ACTC on applications to apply both offline and online protections to applications.

On the ASPIRE website, we have added a new page that contains links to the open source repositories. We have written documentation on how to set up the Docker container, how to run the ACTC and how to apply offline and online techniques to a demo application. This documentation is based on the *D5.13 ASPIRE Open Source Manual Deliverable*.

Furthermore, we have open sourced and documented the ADSS Full at https://github.com/SPDSS/adss. We have also open sourced and documented the ADSS Light at https://github.com/uel-aspire-fp7/adss-light .

1.5.10 Cooperation with other projects

UGent collaborated with the TETRACOM FP7 project (http://www.tetracom.eu) to further prepare its IP for exploitation by an industrial partner, as documented more extensively in deliverable D7.03.

UGent's ASPIRE team also provides input to the vision building processes in the HiPEAC Network of Excellence (http://www.hipeac.net), in particular for drafting the HiPEAC vision documents and roadmaps on compiler technology and their use for protecting and securing software and computer systems.

Furthermore, the project coordinator has been active in the Digital Asset Protection Association (DAPA) project/organization (http://www.digitalassetprotectionassociation.org/), where he has been working with the scientific board members to draft a whitepaper on best practices for publishing and evaluation software protection papers and results. That effort is not finalized yet, however, and unfortunately, as DAPA has become a sleeping organization, this effort is currently stalled.

Within POLITO, there is a collaboration between the ASPIRE researchers and the researchers of the SECURED project. The SECURED project needs to remotely attest the software that has to execute the user security applications.

Nagravision participates to the Celtic-plus project (http://celticplus.eu/) on HEVC Hybrid Broadcast Video Services (H2B2VS, http://h2b2vs.epfl.ch). That project investigates the hybrid distribution of TV programs and services over heterogeneous networks.

1.5.11 The ASPIRE Consortium

ASPIRE was an FP7 collaborative research project that brings together three market leaders in security ICT solutions and four academic institutions from 6 European countries. Gemalto SA (FR) is the world leader in the smart card business. SFNT GmbH (DE) is the world leader in token-based software licensing[1]. Nagravision SA (CH) is the world's leading supplier of end-to-end security solutions for set-top box TV operators. Combined, these three companies understand the varying requirements of security solutions in the diverse markets that need such solutions. Ghent University, Politecnico di Torino, Fondazione Bruno Kessler and University of East London provide the necessary expertise in state-of-the-art software protection techniques and tool chains that cover offline as well as online techniques. They also provide extensive expertise in evaluation methodologies and metrics for software protection.

NGANG B
VAKGROEP ELEKTRONICA EN INFORMATIESYSTEMEN (ELIS)
VAKGROEP TELECOMMUNICATIE EN INFORMATIEVERWERKING (TELIN)
VAKGROEP INFORMATIETECHNOLOGIE (INTEC)

Figure 2: The ASPIRE Consortium at its kick-off meeting in Gent (Nov 2013)

[1] After the start of the project, SFNT GmbH was acquired by Gemalto SA.

Section 2 Use and Dissemination of Foreground

2.1 Dissemination Measures (public)

Dissemination represents a key part within any research project since the awareness and publicity of a project is important to ensure the project success.

A list of all scientific (peer reviewed) publications relating to the foreground of the project as well as a list of all dissemination activities (publications, conferences, workshops, web sites/applications, press releases, flyers, articles published in the popular press, videos, media briefings, presentations, exhibitions, thesis, interviews, films, TV clips, posters) is provided below. These tables are cumulative which means that they show all publications and activities from the beginning until after the end of the project.

More information about the dissemination activities in the project can be found in the public Deliverable D7.05 Dissemination Report.

2.1.1 List of scientific (peer reviewed) publications (public)

With respect to the open access column in the table below, we note that we confirmed open access availability if the publication is accessible via the author's personal websites or via their institutes' repositories.

TEMPLATE A1: LIST OF SCIENTIFIC (PEER REVIEWED) PUBLICATIONS, STARTING WITH THE MOST IMPORTANT ONES										
NO.	Title	Main author	Title of the periodical or the series	Number, date or frequency	Publisher	Place of publication	Year of publication	Relevant pages	Permanent identifiers[2] (if available)	Is/Will open access[3] provided to this publication?
1	Automatic Generation of Opaque Constants Based on the K-clique Problem for Resilient Data Obfuscation	Roberto Tiella, Mariano Ceccato	IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER-2017)	21-24 Febr 2017	IEEE	Klagenfurt, AU	2017			yes
2	A Tightly-Coupled Self-Debugging Software Protection	Bert Abrath, Joris Wijnant, Bart Coppens, Bjorn De Sutter, and Stijn Volckaert	International Workshop on Software Security, Protection and reverse Engineering (SSPREW-2016)	5-6 Dec 2016	ACM	Los Angeles, US	2016	7:1-7:10	https://doi.org/10.1145/3015135.3015142	Yes

[2] A permanent identifier should be a persistent link to the published version full text if open access or abstract if article is pay per view) or to the final manuscript accepted for publication (link to article in repository).
[3] Open Access is defined as free of charge access for anyone via Internet. Please answer "yes" if the open access to the publication is already established and also if the embargo period for open access is not yet over but you intend to establish open access afterwards.

3	Static Analysis and Penetration Testing from the Perspective of Maintenance Teams	Mariano Ceccato, Riccardo Scandariato	ACM/IEEE International Symposium on Empirical Software Engineering and Measurements ESEM 2016	8-9 Sept. 2016	ACM	Ciudad Real, ES	2016	25:1-25:6	https://doi.org/10.1145/2961111.2962611	Yes
4	Search Based Clustering for Protecting Software with Diversified Updates	Mariano Ceccato, Paolo Falcarin, Alessandro Cabutto, Yosief Weldezghi Frezghi, Cristian-Alexandru Staicu	Symposium on Search-Based Software Engineering SSBSE 2016	8-10 Oct 2016	Springer	Raleigh, North Carolina, US	2016	159-175	https://doi.org/10.1007/978-3-319-47106-8_11	Yes
5	Reactive attestation. Automatic detection and reaction to software tampering attacks	Alessio Viticchié, Cataldo Basile, Andrea Avancini, Mariano Ceccato, Bert Abrath, Bart Coppens	ACM Workshop on Software PROtection (SPRO 2016)	24-28 Oct 2016	ACM	Vienna, AU	2016	91-92	https://doi.org/10.1145/2995306.2995315	Yes
6	Towards Automatic Risk Analysis and Mitigation of Software Applications	Regano, Leonardo; Canavese, Daniele; Basile, Cataldo; Viticchié, Alessio; Lioy,	10th IFIP WG 11.2 International Conference, WISTP 2016	26-27 Sep 2016	Springer	Heraklion (GR)	2016	120-135	https://doi.org/10.1007/978-3-319-45931-8_8	yes
7	Assessment of data obfuscation with residue number coding	Biniam Fisseha Demissie, Mariano Ceccato, Roberto Tiella	IEEE/ACM International Workshop on Software Protection	19 May 2015	IEEE	Firenze, IT	2015	38-44	https://doi.org/10.1109/SPRO.2015.15	Yes
8	Automatic Discovery of Software Attacks via Backward Reasoning	Cataldo Basile, Daniele Canavese, Jerome D'Annoville, Bjorn	IEEE/ACM International Workshop on Software Protection	19 May 2015	IEEE	Firenze, IT	2015	52-58	https://doi.org/10.1109/SPRO.2015.17	Yes

		De Sutter, Fulvio Valenza								
9	A reference architecture for software protection	Bjorn De Sutter, Paolo Falcarin, Brecht Wyseur, Cataldo Basile, Mariano Ceccato, Jerome d'Annoville, Michael Zunke	Working IEEE/IFIP Conference on Software Architecture. (WICSA)	4–8 Apr 2016	IEEE	Venice (IT)	April 2016	291-294	https://doi.org/10.1109/WICSA.2016.43	No
10	Software Protection with Code Mobility	Alessandro Cabutto, Paolo Falcarin, Bert Abrath, Bart Coppens, Bjorn De Sutter	Workshop on Moving Target Defense MTD@CCS 2015	12-16 Oct 2015	ACM	Denver, US	2015	95-103	https://doi.org/10.1145/2808475.2808481	Yes
11	Assessment of Source Code Obfuscation Techniques	Alessio Viticchié, Leonardo Regano, Marco Torchiano, Cataldo Basile, Mariano Ceccato, Paolo Tonella, Roberto Tiella	IEEE International Working Conference on Source Code Analysis and Manipulation SCAM 2016	2-3 Oct 2016	IEEE	Raleigh, North Carolina, US	2016	11-20	https://doi.org/10.1109/SCAM.2016.17	Yes
12	Attack simulation based software protection assessment method (Best Paper Award)	Gaofeng Zhang, Paolo Falcarin, Elena Gómez-Martínez, Shareeful Islam, Chris-	International Conference On Cyber Security and Protection of Digital Services (Cyber Security 2016)	13-14 Jun 2016	IEEE	London, UK	2016	1-8	https://doi.org/10.1109/Cyber-SecPODS.2016.7502352	Yes

		tophe Tartary, Bjorn De Sutter, Jerome D'Annoville								
13	Attack Simulation based Software Protection Assessment Method with Petri Net	Gaofeng Zhang, Paolo Falcarin, Elena Gómez-Martínez, Shareeful Islam, Christophe Tartary, Bjorn De Sutter, Jerome D'Annoville	International Journal on Cyber Situational Awareness	Vol 1:1, ISSN 2057-2182	Centre for Multidisciplinary Research, Innovation and Collaboration		2016	8 pages	http://www.c-mric.org/ijcsa/article8.pdf	Yes
14	A Measurement Framework to Quantify Software Protections (Poster + Extended Abstract)	Paolo Tonella, Mariano Ceccato, Bjorn De Sutter, Bart Coppens	ACM SIGSAC Conference on Computer and Communications Security	3-7 Nov 2014	ACM	Scottsdale, Arizona (US)	2014	1505-1507	https://doi.org/10.1145/2660267.2662360	Yes
15	Towards a Unified Framework for Evaluating the Strength of Software Protections (Extended Abstract)	Bjorn De Sutter	ARO Workshop on Continuously Upgradeable Software Security and Protection	7 Nov 2014		Scottsdale, Arizona (US)	2014	34-35		No
16	Reflections on Software Renewability from an Industry Perspective (Extended Abstract)	Brecht Wyseur	ARO Workshop on Continuously Upgradeable Software Security and Protection	7 Nov 2014		Scottsdale, Arizona (US)	2014	36-37		No
17	On the Need for More Human Studies to Assess Software Protection (Extended Abstract)	Mariano Ceccato	ARO Workshop on Continuously Upgradeable Software Security and Protection	7 Nov 2014		Scottsdale, Arizona (US)	2014	55-56		Yes

2.1.2 List of dissemination activities (public)

Possible types of audience mentioned in the table: a) Scientific Community (higher education), b) Industry, 3) Civil Society, 4) Policy Makers, 5) Media

Nr	Type of Activities	Main Leader	Title	Day	Place	Type of audience a)	b)	c)	d)	e)	Size of Audience	Nat/Int
1	Workshop	FBK	Internal seminar	17/10/2013	Trento, IT	x					20	Nat
2	Presentation	NAGRA	NAGRA internal presentation	22/10/2013	Cheseaux, CH		x				10	Nat
3	Workshop	FBK	Internal seminar	24/10/2013	Trento	x					20	Nat
4	Web	UGENT	Website/logo available	1/11/2013	online	x	x	x	x	x	N/A	Int
5	Press Release	UGENT	ASPIRE startup announcement Belgian Press	4/11/2013	online			x	x	x	N/A	Nat
6	Interview	UGENT	Interview with CO on local television station AVS	5/11/2013	online		x	x	x	x	thousands	Nat
7	Presentation	NAGRA	NAGRA internal presentation	28/11/2013	Cheseaux, CH		x				50	Int
8	Web	UEL	ASPIRE project advertised on UEL website: 460.000 EUR to develop software protection	1/12/2013	online	x	x	x	x	x	N/A	Int
9	Presentation	POLITO	ASPIRE: Advanced Software Protection: Integration, Research, and Exploitation	5/12/2013	Torino, IT	x					150	Nat
10	Web	GTO	Wiki page on the Intranet describing the project and expected results	6/12/2013	online		x				whole company	Int
11	Presentation	POLITO	ASPIRE: Advanced Software Protection: Integration, Research, and Exploitation	9/12/2013	Torino, IT	x					100	Nat
12	Flyer	UGENT	Official ASPIRE-leaflet available on the website	15/01/2014	online	x	x	x	x	x	N/A	Int

13	Presentation	UGENT	Keynote speech at CS2 workshop collocated with HiPEAC conference: A Golden Standard for Evaluating Software Protection against Man-at-the-End Attacks	20/01/2014	Vienna, AU	x	x				25	Int
14	Presentation	UEL	Software Protection research overview	20/01/2014	London, UK	x		x			10	Nat
15	Presentation	UEL	Software Protection Research Overview	20/01/2014	London, UK	x					20	Nat
16	Presentation	UEL	Software protection seminar - ASPIRE overview	2/12/2014	Torino, IT	x					20	Nat
17	Presentation	POLITO	ASPIRE: Advanced Software Protection: Integration, Research, and Exploitation	26/02/2014	Torino, IT	x					15	Nat
18	Thesis	FBK	Code Diversity: Code Obfuscation and Clustering Heuristic to Prevent Code Tampering	12/03/2014	Trento, IT	x					30	Nat
19	Web	NAGRA	ASPIRE Project advertised on Nagravision Intranet	13/03/2014	Cheseaux, CH		x				3000+	Int
20	Presentation	POLITO	ASPIRE: Advanced Software Protection: Integration, Research, and Exploitation	14/03/2014	Torino, IT	x					20	Nat
21	Presentation	UEL	Software Protection overview	21/03/2014	London, UK		x	x			20	Nat
22	Exhibition	NAGRA	EU CyberSecurity Strategy - High-Level Conference	28/03/2014	Brussels, BE	x	x	x	x	x	480	Int
23	Press Release	UGENT	Final version of Press Release sent out to the partners	17/04/2014	online						7	Int
24	Press Release	UGENT	Press Release published on CORDIS Wire	18/04/2014	online	x			x		N/A	Int
25	Press Release	UGENT	Press Release published on Alpha Galileo	25/04/2014	online	x	x	x			4000	Int

26	Press Re-lease	UGENT	Press Release sent to our contacts at ACM Tech News	25/04/2014	online						N/A	Int
27	Press Re-lease	UGENT	Press Release published by ACM Tech News	25/04/2014	online	x	x	x	x	x	100000	Int
28	Presentation	UEL	UEL Expert Series: ASPIRE	30/04/2014	London, UK	x					20	Int
29	Interview	UGENT	Interview by Olga Gadyatskaya from the SECCORD project	30/04/2014	online	x					1	Int
30	Press Re-lease	UGENT	Short Press Release sent to: European Cyber Security Round Table	7/05/2014	online	x			x		N/A	Int
31	Press Re-lease	UGENT	Short Press Release published by European Cyber Security Round Table in their Newsflash	7/05/2014	online	x			x		N/A	Int
32	Publication	UGENT	Making Advanced Software Protection Tools Usable for Non-Experts	19/05/2015	Firenze, IT	X					N/A	Int
33	Conference	POLITO	Annual Privacy Forum 2014	20-21/05/2014	Athens, GR	x	x	x	x	x	TDB	Int
34	Conference	POLITO	Cyber Security & Privacy Forum 2014	21-22/05/2014	Athens, GR	x	x	x	x	x	TDB	Int
35	Conference	UGENT	Evaluating the Strength of Software Protections	25-30 May 2014	Dagstuhl, DE	X					50	Int
36	Conference	FBK	36th International Conference on Software Engineering	31/05/2014	Hyderabad, IN	x	x				1200	Int
37	Presentation	UGENT	Evaluating the strength of software protections	11/06/2014	Dagstuhl, DE	x	x				44	Int
38	Publication/Thesis	FBK	Code Diversity: Code Obfuscation and Clustering Heuristic to Prevent Code Tampering	June 2014	Trento, IT	x					N/A	Int
39	Publication/Thesis	FBK	Implementation and Assessment of Data Obfuscation for C/C++ Code Based on Residue Number Coding	June 2014	Trento, IT	x					N/A	Int

40	Press Re-lease	UGENT	FP7 ASPIRE Project	13/07/2014	Gent, BE	x	x					Int
41	Presentation	NAGRA	White-Box Cryptography	28/07/2014	Verona, IT	x	x				35	Int
42	Interview	UGENT	Scuola anti pirati	30/07/2014	Verona, IT	x	x				N/A	Nat
43	Presentation	UGENT	Evaluating the strength of software pro-tections	30/07/2014	Verona, IT	x	x				35	Int
44	Presentation	UGENT	The ASPIRE project	1/08/2014	Verona, IT	x	x				35	Int
45	Presentation	NAGRA	NAGRA internal security workshop	10/09/2014	Cheseaux, CH		x				50	Int
46	Presentation	NAGRA	NAGRA use case presented during in-ternal workshop	11/09/2014	Cheseaux, CH		x				20	Nat
47	Presentation	FBK	Research Seminar: A Study on the Ef-fect of Code Obfuscation: Quality of code and Efficiency of attacks	23/09/2014	Luxembourg, LU	x					30	Nat
48	Thesis	FBK	Implementation and Assessment of Data Obfuscation for C/C++ Code Based on Residue Number Coding	14/10/2014	Trento, IT	x					50	Nat
49	Conference	FBK/UGENT/ NAGRA	2014 ACM SIGSAC Conference on Computer and Communications Secu-rity	3-7:11/2014	Scottsdale, Arizona, US	x	x				250	Int
50	Workshop	FBK/UGENT/ NAGRA	ARO Workshop on Continuously Up-gradeable Software Security and Pro-tection	7/11/2014	Scottsdale, Arizona, US	x	x				50	Int
51	Presenation	UEL	Software protection seminar - ASPIRE overview	2/12/2014	Milan, IT	x					50	Nat
52	Presentation	UEL	Software protection seminar - ASPIRE overview	2/12/2014	Milan, IT	x					10	Nat
53	Thesis	POLITO	Modellazione di protezioni software attraverso ontologie formali	12/12/14	Turin, IT	x					50	Nat

54	Poster	UGENT	ASPIRE Project Presentation at HiPEAC2015 Conference	19-21/01/2015	Amsterdam, NL	x	x				600	Int
55	Thesis	POLITO	Attestazione remota del software	13/03/15	Torino, IT	x					50	Nat
56	Presentation	UGENT	System Software Lab Research Overview	21/04/2015	Gent, BE	x					40	Nat
57	Exhibition	UGENT	ASPIRE Booth and Poster at the Cybersecurity & Privacy Innovation Forum	28-29/04/2015	Brussels, BE	x	x		x		400	Int
58	Presentation	UGENT	Making Advanced Software Protection Tools Usable for Non-Experts	18/05/2015	Firenze, IT	x	x				43	Int
59	Conference	POLITO	1st International Workshop on Software PROtection (SPRO 2015)	19/05/2015	Florence, IT	x	x				40	Int
60	Publication	FBK	Assessment of data obfuscation with residue number coding	19/05/2015	Florence, IT	x	x				40	Int
61	Presentation	UGENT	ASPIRE project presentation	8/06/2015	Gent, BE	x					1	Int
62	Thesis	UGENT	Automatische injectie van flexibele opake predicaten	23/06/2015	Ghent, BE	x					Public thesis	Nat
63	Thesis	UGENT	SAD Droid: Zelf-Anti-Debugging voor Android	4/09/2015	Ghent, BE	x					Public thesis	Nat
64	Presentation	FBK	ASPIRE - Trustworthy software execution on untrusted mobile platforms	10/09/2015	Luxembourg, LU	x					30	Nat
65	Conference	UEL/UGENT	CCS 2015	12-16/10/2015	Denver, US	x	x				20	Int
66	Workshop	UEL/UGENT	Second Workshop on Moving Target Defense MTD	15/10/2015	Denver, US	x	x				20	Int
67	Press Release	NAGRA	NAGRA Internal Press Release: ASPIRE	16/10/2015	Cheseaux, CH	x	x				3500	Int
68	Presentation	NAGRA	White-Box Cryptography and Smart Cards: Friend or Foe?	15/11/2015	Bochum, DE	x	x				100	Int
69	Thesis	POLITO	Attestazione remota basata su controllo di invarianti	1/12/2015	Torino, IT	x	x				50	Nat

70	Poster	UGENT	ASPIRE Poster Presentation at the HiPEAC2016 Conference	18-20/01/2016	Prague, CZ	x	x				650	Int
71	Workshop	NAGRA	NAGRA-EDSI Exploitation worrkshop	23/02/2016	Rennes, FR		x				10	Int
72	Poster	UGENT	ASPIRE Tool Demonstration	8/03/2016	Cheseaux, CH	x	x				100	Nat
73	Thesis	FBK	An Experimental Study on Run-Time Overhead Introduced by Data Obfusca-tion Transformations	24/03/2016	Trento, IT	x					50	Nat
74	Conference	UEL	A reference architecture for software protection	7/04/2016	Venice, IT	x	x				30	Int
75	Presentation	UEL	Software Protection seminar	20/05/2016	Hangzhou, CH	x					60	Nat
76	Presentation	UGENT	ASPIRE project presentation	8/06/2016	Gent, BE	x					1	Int
77	Conference	UEL	Attack Simulation based Software Pro-tection Assessment Method for Protec-tion Optimisation at the Cypber Secu-rity 2016	14/06/2016	London, UK	x	x				50	Int
78	Poster	UEL	Poster: Software protection assess-ment with code metrics and petri nets	16/06/2016	London, UK	x	x	x	x		100	Nat
79	Interview	UGENT	Written interview for security special feature of HiPEACInfo48: Locking The Back Door	29/06/2016	online	x	x				500	Int
80	Publication, Thesis	FBK	An Experimental Study on Run-Time Overhead Introduced by Data Obfusca-tion Transformations	June 2016	Trento, IT	x					N/A	Int
81	Poster	UEL	Poster: A Light Process for the Soft-ware Protection Assessment Based on Petri Nets	1/07/2016	Valencia, ES	x					100	Nat
82	Presentation	NAGRA	Talk at WhibOx workshop	14/08/2016	Santa Bar-bara, Califor-nia, US	x	x	x	x	x	90	Int

83	Conference	FBK	Static Analysis and Penetration Testing from the Perspective of Maintenance Teams	8/09/2016	Ciudad Real, ES	x					100	Int
84	Interview	UGENT	Combined protections for greater mobile app security	9/09/2016	online	x	x	x	x	x	N/A	Int
85	Video	UGENT	22 Youtube video's posted about the ASPIRE project and its research results	Sept - Dec/2016	online	x	x	x	x	x	N/A	Int
86	Workshop	POLITO	Towards Automatic Risk Analysis and Mitigation of Software Applications	26/09/2016	Heraklion, Crete, GR	x					50	Int
87	Interview	UGENT	Written interview for Cordis website: Towards Automatic Risk Analysis and Mitigation of Software	26/09/2016	Gent, BE						N/A	Int
88	Conference	FBK/POLITO	Assessment of Source Code Obfuscation Techniques	2/10/2016	Raileigh, NC, US	x					100	Int
89	Conference	FBK/UEL	Search Based Clustering for Protecting Software with Diversified Updates	8/10/2016	Raileigh, NC, USA-	x					100	Int
90	Presentation	UGENT	ASPIRE: Advanced Software Protection: Integration, Research, and Exploitation	18/10/2016	Gent, BE	x	x				50	Nat
91	Presentation	UGENT	ASPIRE: Advanced Software Protection: Integration, Research, and Exploitation	18/10/2016	Gent, BE	x	x	x			50	Nat
92	Publication	UGENT	The ASPIRE Framework for Software Protection.	24-28/10/2016	Gent, BE	x					N/A	Int
93	Press Release	UGENT	Successful ASPIRE SPRO workshop	24-28/10/2016	Gent, BE	x	x	x	x	x	N/A	Int
94	Films	FBK/POLITO	Reactive Attestation: Automatic Detection and Reaction to Software Tampering Attacks	28/10/2016	Vienna, AU	x	x				50	Int
95	Workshop	UGENT	25nd International Workshop on Software Protection (SPRO 2016)	28/10/2016	Vienna, AU	x	x				50	Int

96	Exhibition	UGENT	The ASPIRE Framework for Software Protection	28/10/2016	Vienna, AU	x	x				50	Int
97	Presentation	UEL	ASPIRE: Advanced Software Protection: Integration, Research, and Exploitation	31/10/2016	London, UK	x					25	Nat
98	Conference	UGENT	Tightly-Coupled Self-Debugging Software Protection	6/12/2016	LA, US	x					N/A	Int
99	Conference	FBK	24th International Conference on Software Analysis, Evolution and Reengineering (SANER-2017)	24-24/02/2017	Klagenfurt, AU	X					N/A	Int