PASR

Preparatory Action on the enhancement of the European industrial potential in the field of Security research

Grant Agreement no. SEC5-PR-104600

Robin

Open Robust Infrastructures

Project

D.8 Final Activity Report

Period covered: from February 2006 to May 2008
Date of preparation: 15/05/2008

Start date of Activity: 01/02/2006 Duration: 2.25 years

Activity coordinator name: Hermann Härtig
Activity coordinator organisation name: Technische Universität Dresden Revision 1

CLASSIFICATION: Public

D.8 ROBIN - Final Activity Report

Marcus Völp
Technische Universität Dresden
Department of Computer Science
Operating Systems Group

Stephan Courcambeck
STMicroelectronics
Advanced Systems Technology Group
Rousset, France

Christian Schwarz
STMicroelectronics
Advanced Systems Technology Group
Rousset, France

May 13, 2008

Abstract

ROBIN is a project in the 2005 EC call on Preparatory Actions for Security Research (PASR-2005 Project no. 104600). In this final report we summarise the work performed and the progress achieved.

Our mission is to research and develop an operating-system (OS) infrastructure capable of protecting against the threats our information and communication infrastructure is currently facing. Among the threats about which we are conserned are attacks by hackers and terrorists with the ultimate goal to obtain valuable information (such as industrial secrets) and to shut down critical information and communication infrastructure. Such an attack may exploit viruses and Trojan horses and it may be performed in preparation of an accompanying physical attack.

Following the premise that the mere size and complexity of today's commonly-used legacy OSs such as Linux (Unix) and Windows makes it absolutely impossible to sufficiently harden these systems, we have developed an operating-system infrastructure that supports critical applications without having to rely on such complex operating systems. Instead, we have developed a microhypervisor-based operating system that offers a minimal set of components sufficient to run the critical parts of an application. We call this set the *trusted-computing base* (TCB) of this application. Through the use of virtual-machine (VM) technology one can still enjoy the comfort and look and feel of legacy operating systems for the uncritical parts of an application and while the system operates normally. We have carefully designed and implemented a VM technology to be not part of the TCB of the critical parts of an application. Under an attack or in an overload situation we fall back to a possibly less comfortable mode of operation that reliably provides all functionality required during a case of an emergency.

In the following report we detail this operating-system infrastructure. We exemplify our approach with two application scenarios: the placement of an emergency call and secure document processing. Furthermore, we explain how a small part of the microhypervisor — the most critical component — can be formally verified. Once applied to the entire microhypervisor and to all TCB components, this verification has the perspective of providing the highest form of confidence that this technology is reliable.

Contents

1. Final Activity Report

1.1. Activity Execution

ROBIN is focused on researching and developing an operating-system infrastructure capable of protecting against the threats our information and communication infrastructure is currently facing. These threats include not only direct attacks on the communication infrastructure in embassies, border control posts, police stations or other governmental organizations, but also break downs of these devices due to overload as a side effect of disasters and physical attacks.

For an illustrative example consider the following scenario. A hacker successfully launches some physical attack in parallel to a virus attack against mobile-phone OSs with the objective to shut down all mobile-phone communication in the area of the attack. Precious time is lost until an emergency call can inform the helping organizations.

One might argue that this scenario is unrealistic, however in part this attack has already happened. In August 2000, a saturation attack placed by a mobile-phone worm has successfully shut down the Japanese emergency services number [6].

It is precisely the vulnerability of the commonly used operating systems Linux (Unix), Windows and Symbian that makes such a scenario possible. These systems have become so large and so complex that it is virtually impossible to understand them to a degree that allows harding them enough to withstand such attacks.

It is our main objective to provide an operating-system infrastructure that, at least as far as this kind of critical functionality is concerned, is capable of enduring and surviving these kinds of attacks. We achieved this without having to sacrifice the comfort and look-and-feel of these legacy operating systems by using virtual-machine technology to isolate the critical functionality from potentially vulnerable software.

In the following we detail the work performed, our achievements and how these relate to the state-of-the-art (Section 1.1.3). In Section 1.1.4 we highlight how our solutions improve the security of citizens and how they impact future academic and industrial research. With this final activity report we present our final plan for using the knowledge in Appendix A.1 and the final plan for the classification of the results in Appendix A.2. Next we introduce the beneficiaries that were involved in this project and give a brief overview of the technology we developed.

1.1.1. Beneficiaries Involved

The beneficiaries involved in the ROBIN project are the Operating Systems Group of the Technische Universität Dresden (TUD), the Advanced Systems Technology Groups of STMicroelectronics Rousset and Grenoble (ST), secunet Security Networks AG (secunet) and the Security of Systems Group of the Radboud Universiteit Nijmegen (RU). In the following we give a brief overview over these parties and highlight their responsibilities within the project.

Technische Universität Dresden (TUD)

The Operating Systems Group of the Technische Universität Dresden lead by Prof. Hermann Härtig has a track record in the development of microkernels and microkernel-based system for real-time and security critical application. L4/FIASCO — TUD's implementation of the L4 microkernel interface — is widely used, among others in a hand-held device of an American manufacturer (under a commercial licence [1]). Since 1996 the group maintains a paravirtualized version of the Linux kernel — L4Linux — that runs de-privileged as an isolated user-level application on top of L4. Furthermore, in a joint project, TUD and Nokia are developing a version of the Symbian operating system that runs de-privileged on top of L4.

TUD has joint this project with the strong interest to disseminate microkernel and microhypervisor technology, to increase the acceptance of this technology for security architectures for stationary as well as for embedded systems and to further Europe's security expertise by attracting highly qualified researchers to work in Europe and in Dresden in particular.

TUD's main responsibilities within ROBIN are in the development of the microhypervisor *Nova* (WP1) and of the de-privileged virtual-machine monitor (VMM) required to run fully virtualized legacy operating systems in an isolated fashion (WP1). Furthermore, TUD developed a security architecture called *Bastei* (WP2) which provides a minimal trusted-computing base to the security critical applications and which has tight control of the resources used by the VMM and the legacy operating-systems such that in case of an emergency, sufficient resources can be made available to critical applications.

Nova is a research prototype and possible successor of L4/FIASCO which aggressively targets full virtualization by exploiting the recent hardware extensions such as Intel VT and which is developed having the upcoming multi-core architectures in mind. The source code of Bastei and Nova is respectively will be released in the near future under the terms of an open source license. Furthermore, Bastei will be supported by a recently founded spin-off.

In addition to the above work, a member of the TUD group was involved in the verification of the selected Nova module.

STMicroelectronics (ST)

STMicroelectronics is one of the leading European semiconductor manufacturers. ST designs semiconductors for a range of applications like automotive systems, computer peripherals, mobile phones, set-top-boxes, etc. In the context of the ROBIN project, ST's goal was to demonstrate the feasibility of a ROBIN-like architecture on an embedded system such as a mobile phone. ST already had some experience in the development of an L4-based secure embedded OS in its Grenoble and Rousset affiliates.

Two STMicroelectronics affiliates collaborated to the ROBIN project through the Advanced System Technology (AST) R&D organization: STMicroelectronics Grenoble SAS and STMicroelectronics Rousset SAS.

ST Grenoble The AST Operating Systems and Programming Models (OSPM) team has a strong expertise in Operating System research and development in the context of embedded systems like mobile phones and set-top-boxes. Prior to the ROBIN project, they have collaborated with

[1] The license terms do not allow us to disclose the name of the licensee.

STMicroelectronics Rousset to develop an L4-based secure embedded system for platforms based on the ST231 VLIW processor by fully porting an L4 implementation on the ST231. In the context of the ROBIN project, they concentrated in WP1 on the adaptation of this L4 implementation into a simplified form of hypervisor on a software simulator of a cohosting capable processor. This work allowed us to execute concurrently Linux and an L4-based TCB and secure applications on the simulator with guarantees that Linux (even the kernel) cannot interfere with the minimal TCB and its applications.

ST Rousset The AST Security team of STMicroelectronics Rousset SAS has a strong expertise in security of embedded systems such as mobile phones and set-top-boxes. This team, together with the AST OSPM team in Grenoble, developed an L4-based secure operating system for embedded platforms. The team focused more specifically on the user-level part of the system. In parallel, a formal model of the user-level components was developed using the B formal method. In the context of the ROBIN project, STMicroelectronics Rousset SAS contributed with STGrenoble on the adaptation of L4 into a simplified form of hypervisor on a software simulator of a cohosting capable platform. STRousset was also responsible in WP2 for enhancing the operating system in order to add support for secure graphical user input and output on an embedded platform, and to design and implement the integration of Linux as an L4 task on cohosting capable platforms (including user input and output). Finally, in WP3, ST Rousset was responsible for showing the applicability of the ROBIN architecture to embedded platforms by choosing a demo scenario — the placement of an emergency call — and implementing it on a hardware emulator of an ARM1176-TrustZone-based platform.

secunet Security Networks AG (secunet)

secunet Security Networks AG is one of Europe’s leading suppliers of products and services in the area of highly complex IT security solutions with 10 years of experience in protecting electronic information.

A highly qualified and experienced staff of 230 is working in the business areas of high security (SINA), government and business security and automotive, to develop the optimum solution to suit the unique IT security requirements of secunet’s customers.

The reference list includes the majority of DAX30 companies and major international companies as well as organizations and public authorities both in Germany and abroad.

In this project, secunet used it’s extensive knowledge of customer requirements to analyze and develop scenarios which exploits the operating-system capabilities provided by the ROBIN architecture (WP5). One selected real-life scenario was implemented as a research prototype (WP5).

Radboud Universiteit Nijmegen (RU)

The Security of Systems (SoS) group headed by Prof. Dr. Bart Jacobs is part of the Institute for Computing and Information Sciences at the Radboud University Nijmegen. The group currently consists of 5 permanent members, 3 post-docs, and 8 PhD students.

The origin of the SoS group is the LOOP verification project investigating the formal semantics of object-oriented programming languages and tool-assisted program verification [5].

The group coordinated the EU-IST FP5 project VerifiCard [8] where it applied its expertise on formal specification and verification on the latest generation of smartcards in collaboration with smartcard manufacturers. The project was successfully completed at the end of 2003 and the deliverables were judged to be of excellent quality in the final review.

This line of work is being continued in two national projects funded by the Dutch Technology Foundation STW and one European project. The PINPAS project investigates the security of modern smartcard operating systems, more specifically side-channel attacks; this is a project in cooperation with the universities of Twente and Eindhoven, the Dutch company TNO-ITSEF that specialises in security evaluations, STMicroelectronics Belgium, and Giesecke & Devrient in Germany. The JASON project, in cooperation with Chess-IT, develops a security architecture for ambient applications that involve smartcards and embedded devices.

The FP6 integrated project Mobius [7] investigates a security architecture for global computing based on Proof-Carrying Code (PCC); in this project the SoS group's heads the workpackage on program verification. This project involves several industrial partners, namely France Telecom, SAP, Trusted Logic, and TLS Technologies.

Work on security in the group has been boosted by a big 'Pionier' research grant by the Dutch foundation for scientific research NWO. Here work focuses on the design and verification of security protocols and smart cards. The group maintains a much broader perspective on security; for instance, in a project for the Dutch government the groups developed and formally verified some of the software that was used in an experiment in Internet voting at the last European elections in the Netherlands. Indeed, the group also considers it part of its duty as a publicly funded and independent institute to take active interest in societal aspects of security, such as electronic voting, open source software, and privacy.

Within this project, RU is responsible for the formal specification of the Nova microhypervisor and for the formal verification of a selected Nova kernel module (WP4).

1.1.2. The ROBIN Architecture – a Brief Overview

In the following we present a brief overview of the ROBIN Architecture. A much more detailed description can be found in the advisory board executive summary [17].

The main construction principle applied in the ROBIN architecture (as shown in Figure 1.1) is the isolation of critical and uncritical components through address spaces (respectively virtual machines) as they are provided by the underlying Nova microhypervisor on the PC platform respectively by the cohosting enabled version of L4 on embedded platforms. Like a microkernel, the microhypervisor implements only the functionality that is required to achieve this isolation. Unlike a microkernel, which only implements address spaces and a single isolated legacy OS when run on cohosting enabled hardware, a microhypervisor supports the construction of virtual machines and is thus capable of running multiple unmodified legacy OS concurrently.

The security layer on top of the microhypervisor consists of several in address spaces isolated components that provide the minimally required functionality for security critical applications. These components manage all platform resources except those required by the microhypervisor.

Due to the stringent isolation of these components we were able to achieve a very important property for constructing minimal TCBs: although a component may exist in the security layer, it adds to the trusted computing base of an application only when this application relies on the functionality provided by this specific component. More precisely, a component is only part of the trusted computing base if the functionality it provides is indispensable for the security properties to achieve.

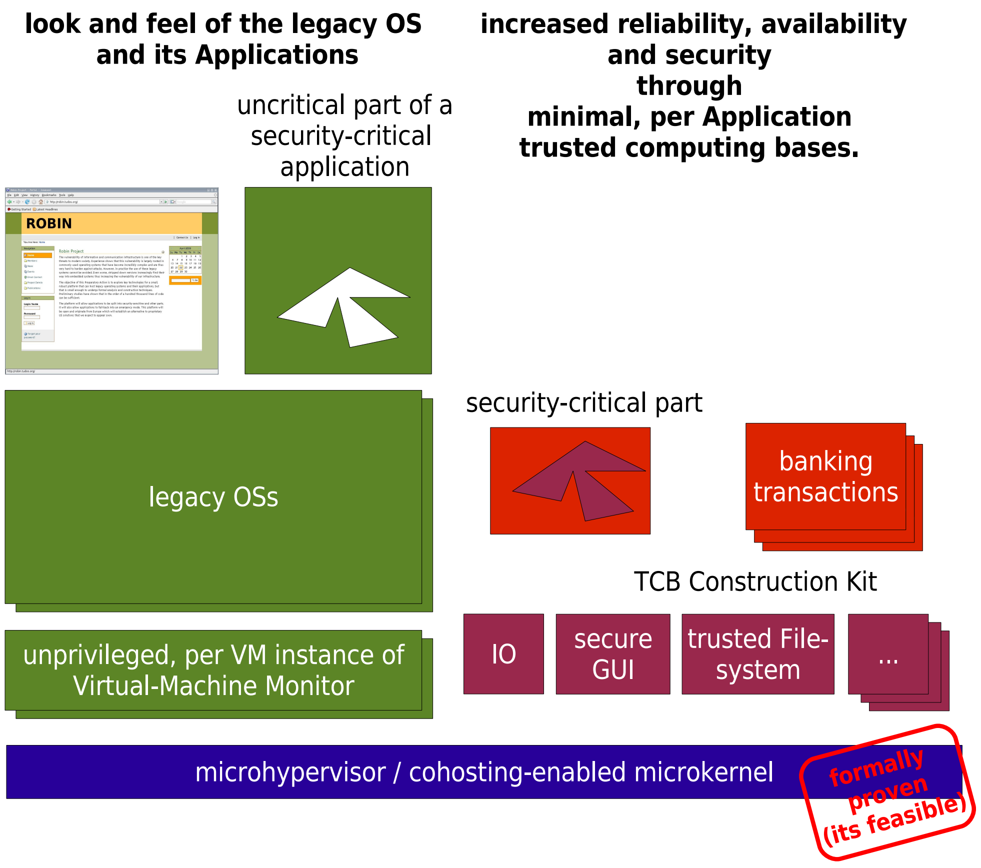

look and feel of the legacy OS and its Applications
increased reliability, availability and security through minimal, per Application trusted computing bases.
uncritical part of a security-critical application
ROBIN
legacy OSs
unprivileged, per VM instance of Virtual-Machine Monitor
security-critical part
banking transactions
TCB Construction Kit
IO
secure GUI
trusted File-system
...
microhypervisor / cohosting-enabled microkernel
formally proven (its feasible)

Figure 1.1.: Construction principles of the ROBIN architecture.

This means we can reuse functionality provided by a legacy operating system and its applications — both of which we assume may successfully be penetrated by an attacker — for critical applications. To give an illustrative example, a secure bank application may store confidential data such as the account information in the file system of an untrusted legacy OS provided this information has been secured by a trusted component through cryptographic means. In this case the legacy OS is not part of the TCB of this application in terms of confidentiality (but it is in the TCB in terms of availability).

An observation that can be made in many security critical applications as well as in applications that must be reliable in emergency situations is that a rather small part of this application must really be dependable. It is this part that we split from the remaining part of the application and which we run on top of the security layer. To be able to use this small part in case of an emergency without having to rely on the remaining larger part of this application we extend the small part of interactive applications with a small and possibly less comfortable user interface. As a consequence of this, we preserve the comfort and look-and-feel of security sensitive applications as long as the legacy OSs work properly. In case of a successful attack to these systems we maintain the critical functionality by falling back to these less comfortable interfaces and to the small security critical parts of the applications.

1.1.3. Work Performed, Achievements and Relation to the State-Of-The-Arts

In the following presentation on the work performed, on the achievements made and on their relation to the state-of-the-art we follow the projects workpackage structure. The work in Robin is organized in the following 6 workpackages:

Workpackage WP1: Microhypervisor

Workpackage WP2: Security Architecture

Workpackage WP3: ROBIN on embedded platform

Workpackage WP4: Formal Verification

Workpackage WP5: Application Scenario

Workpackage WP6: Project Management

All published deliverables are available at http://robin.tudos.org.

WP1: Microhypervisor

The objectives of workpackage WP1 have been to design and implement a small and secure microhypervisor and its embedded counterpart — a cohosting enabled kernel — for embedded systems. The microhypervisor is the only component that belongs to the trusted computing base of every other component. It provides temporal and spatial isolation of components and supports secure message transfers between components. Furthermore, it presents a virtual machine abstraction for running unmodified legacy guest operating systems next to enlightened components of a multiserver system by utilizing new hardware functionality, such as Intel Vanderpool Technology (VT), AMD Pacifica and ARM Trustzone.

TUD:

In this workpackage TUD designed and implemented the Nova microhypervisor for Intel VT processors. Furthermore we developed a deprivileged virtual-machine monitor called Vancouver.

Nova
The design of the microkernel interface was mainly driven by our vision to run a multi-server operating system with isolated components next to virtual machines, to run de-privileged virtual-machine monitors and to enable these components to interact frequently and with a low communication overhead. Although this project does not focus on multiprocessor systems and on real-time systems we payed special attention to the requirements of these systems to not hinder future developments in this direction.

The results of this design work are documented in an architecture whitepaper (D.1) and in the microhypervisor specification (D.2), which will be published in the course of the open-source release of the microhypervisor implementation.

The microhypervisor specification contains in addition to a textual description a description in the form of pseudo code. This pseudo code was used to communicate precise facts about the interface and changes therein to the theoreticians (WP.4) responsible for formalizing this specification. It turned out that this form of communication was very efficient and that it rendered it unnecessary to teach a formal-specification language to the kernel developers.

The complete implementation and testing of this interface is ongoing work which is also the cause why we have to delay the public release of the Nova sources. At the point of writing this document Nova supports the feature set required to run the L4/Fiasco kernel and an unmodified Linux kernel inside a virtual machine whereas the necessary virtual devices that are required to run these systems are provided by the Vancouver VMM.

Accompanying the implementation of the Nova interface we measured the performance of several hardware features and of alternative implementations.

An implementation detail to highlight here is the consequent implementation of Nova as an interrupt style kernel and the use of continuations to introduce certain points at which system calls can be preempted. This is accompanied by an analysis where these preemption points could be placed to achieve similar low interrupt latencies than more preemptible kernels. To our knowledge Nova is the first kernel for which such an analysis has been performed.

A detailed description of the Nova implementation is provided as deliverable D.9.

Vancouver

The Vancouver VMM runs as a de-privileged user-level program on top of the Nova microhypervisor. Its task is to establish the execution environment for the virtual machines and in this course to handle all but the most performance critical virtualization events — i.e., vTLB faults — provided by the Intel VT hardware and forwarded by the Nova microhypervisor. In a virtual-machine monitor, the vTLB is responsible for the construction of page tables that establish a direct mapping between the memory addresses of an application running inside the VM and the respective addresses in physical memory. In a virtual-machine setting, the information required for this mapping is only available as two related mappings. The first — the page-tables of the guest operating system — define the mapping between guest virtual to guest physical addresses, the components involved in the memory management for this VM together establish the second mapping between guest physical and host physical addresses.

To establish the execution environment as expected by the guest operating system, Vancouver contains an instruction emulator which is used to execute code of processor modes for which Intel VT hardware provides no hardware support (e.g., the x86 real mode). An important observation that helped us reduce the size and complexity of this component (by at least an order of magnitude compared to bochs and qemu) that in our setting we can execute most instructions directly after we have verified and possibly adjusted their memory references. Vancouver currently implements device models for PS2, keyboard, VESA, a subset of PCI. Furthermore, we made a feasibility analysis of network card virtualization and identified the Intel e1000 card as a promising target.

Like Nova, the implementation of Vancouver is ongoing work. At the time of writing this document an unmodified Linux kernel can be run inside a virtual machine.

STRousset and STGrenoble: On an internal simulator of a cohosting capable processor, ST Grenoble tested and adapted the modifications in L4 allowing it to execute on the secure side of the processor, while the Linux kernel executes on the non-secure side. The adaptations to the ARM1176-TrustZone processor, originally planned for development at ST Grenoble, were performed out of the scope of the ROBIN project.

1) TrustZone
TrustZone-like hardware modifications [2] consist in modifying a processor to duplicate the classical User and Kernel modes. This results in the possibility for the processor to execute in 4 different modes: Non-Secure User, Non-Secure Kernel, Secure User and Secure Kernel. The two secure modes are also referred to as the ”secure world”, and the two non-secure modes as the ”non-secure world”. The secure world has access to all hardware resources (memory, peripherals, etc), while the non-secure world has limited visibility on resources, configured by the software executing in the secure kernel mode (thanks to different sets of virtual page tables, and configurable hardware logic to access peripherals).

2) TrustZone extensions to L4
The modifications had to allow execution of the Linux kernel and its applications with minimal changes in the L4 kernel and the best possible integration in the L4 environment. These original modifications consisted in creating a new ”ghost” L4 thread to represent the complete software stack running on the non-secure side. The L4 kernel, executing on the secure side, uses a secure timer and the regular L4 scheduling to decide when to execute this thread. When this thread is scheduled, the L4 kernel uses a dedicated (new) instruction to cause the switching of the processor into the non-secure side.

WP2: Security Architecture

The objectives of this workpackage were the design and implementation of the ROBIN trusted computing base toolkit. The focus of this toolkit is in enabling per application trusted computing bases consisting of a set of secure servers. The servers of this platform implement the basic functionality of the ROBIN architecture. The focus thereby is on providing small, dependable servers implementing the security sensitive functionality.

TUD:

Based on the lessons learned from L4env, a multi-server operating system running on top of L4, TUD has designed and implemented the Bastei trusted-computing base toolkit on the PC platform. This work is documented in much more detail in the Bastei Architecture Whitepaper D.3, in the Report on the Bastei Demonstrator D.10 and in the Bastei Interface Specification D.4.

Bastei has been designed and implemented around the concept of distributed, hierarchical security-policy managers. Each security-policy manager manages the local name space of its subsystem and can therefore enforce both access- and information-flow policies. In addition, the security-policy manager are capable of selectively revoking the resources servers have allocated on behalf of a component in its subsystem. It can therefore faithfully shutdown individual components or even entire subsystems, for example, a possibly penetrated legacy OS and its corresponding instance of the Vancouver virtual-machine monitor. Despite all these possible policy decisions, Bastei maintains a performance comparable to L4env. This is because of the consequent application of the Session concept which supports a direct communication between clients and their respective servers. Policy enforcement happens at session establishment, however, security-policy managers may shut-down sessions at any later point in time.

Two more highlights to mention about Bastei are its dynamic RPC framework, which builds on the fact that Bastei has been implemented in C++ and completely object oriented, and its parallel development on L4 and Linux.

Dynamic RPC [4] is an IOstream-based framework used to assist in the encoding of the communication protocols between clients and servers. Because DynRPC is implemented in C++ no additional interface description language (IDL) is required which possibly adds hidden complexity to servers. Also there is no immediate need for an IDL compiler.

In addition to an implementation on top of the L4 microkernel and on top of the Nova microhypervisor, the core Bastei servers have also been implemented as a Linux application program. This way, programmers developing security sensitive applications and new Bastei components can reap benefit of the tool support available for Linux application development and later on run these application on top of L4 and Nova for increased security properties.

At the time of writing this report all core Bastei components have been implemented and succeeded against first penetration tests. Among these is a secure graphical user interface which is a central component of the Bastei Demonstrator (D.10).

In addition to these core components, TUD developed an L4env legacy container that allows us to run L4Linux on top of Bastei and first native drivers we used in the interactive demonstrator.

ST Rousset: The work performed by STRousset in WP2 consisted in 2 parts.

The first part consisted in extending the minimal L4-based TCB to integrate the support for Linux and its applications in such a way that they can really be seen as a regular L4 application executing on top of L4. This was designed and implemented in such a way that Linux and its applications may use the TCB services the same way as L4 applications do, and are subject to the same security policy as L4 applications.

The second part consisted in extending the minimal TCB with support for graphical user input and output. This was achieved by integrating the DoPE Graphical User Interface from TUD on top of the minimal TCB on the secure side. Besides, a Linux Frame Buffer (FB) driver has been designed and implemented. This driver transforms FB events and requests into cross-world messages to/from the DoPE user-level server. During Linux startup, this driver registers as a DoPE client, and allocates and shares a memory area used as a frame buffer. A new DoPE graphical window is created where the content of the frame-buffer memory area is displayed. In parallel, user keyboard inputs into this graphical window are forwarded by DoPE to the Linux FB driver using the FB API.

WP3: ROBIN on embedded platform

The objectives of this workpackage were to demonstrate the ROBIN security architecture is applicable to embedded platforms and to research how embedded applications can be split into critical and non-critical parts.

ST Rousset: The first part of the work in this package consisted in selecting a typical embedded application in order to show that it could be split between a non-critical part that could be left under the control of the OS running on the non-secure side, and a critical part that only depends on the minimal L4-based TCB.

Different alternatives were studied, among which one was chosen as the best demonstrator for embedded platforms such as mobile phones: the placing of a phone call.

In this application, the user address book functionality was left on the rich OS as this is a typically complex application, where most important are the user friendliness, the integration with messaging services, calendars, etc. The nice look and feel and the integration of many services comes at the price of the complexity and potential number of bugs and stability [3].

However, what is critical is really the possibility to dial a phone number, for example in case of an emergency situation, even if the rich operating system is under attack or crashed.

The challenge then was to demonstrate the feasibility of such a split functionality, on a ROBIN-like architecture with minimal changes to the address-book application.

This demonstration was delivered at milestone M.1 on a software simulator of a modified ST processor for embedded systems using a classical Linux console mode address book called abook. The secure graphical user interface running in the secure world was used to allow the user to both check the number to be called (and additional properties such as the type of number: free, toll-number, foreign, etc) and also to type a number to be called in case the address book and/or Linux was crashed or under reboot.

The second and most ambitious part of this workpackage consisted in building a demonstrator of a ROBIN-like architecture on an hardware embedded platform.

In order to achieve this, STRousset had to use a hardware emulator (in Field Programmable Gate Array (FPGA)) of an ARM1176-TrustZone based platform, since no ARM1176-based development platform was available in the time frame of the project. This hardware emulator consists of 5 stacked FPGA boards in which a ARM1176 core, L2 cache, UART, LCD controller, DMA, RTC, and timers were mapped. This development board is aimed at mimicking a TrustZone-based mobile phone platform. The ARM1176-TrustZone is the core of ST's STn8820 Nomadik platform.

In order to have a self-sufficient graphical hardware demonstrator, we had to integrate to the platform a LCD controller hardware block and develop the required driver library as required by the DoPE graphical user interface. STRousset also had to adapt and enhance the communication interfaces between Linux running on the non-secure side and the DoPE graphical user interface running in the secure side. A generic and modular communication infrastructure was developed and integrated in Linux and in the minimal TCB, minimizing the amount of software executing both in the Linux kernel and in the L4 kernel (following the least privilege principle and the microkernel philosophy).

Finally, the secure dialer application developed for the M1 deliverable was adapted to this hardware emulator and to the communication protocol.

In order to cope with the very low speed of the emulator (12MHz, compared to 500MHz for the final chip), we had to stick to a console mode address book in Linux. This sacrifice in terms of look and feel allows us to have a demonstration that can be shown in real time to end users on the real emulator.

This work resulted in the availability on the ARM1176-TrustZone-based hardware emulator of a demonstrator of the ROBIN architecture, including a graphical user interface, on which STRousset was capable of running the demonstrator of a split functionality.

WP4: Formal Verification

The objectives of this workpackage were to develop, evaluate and assess formal methods for the specification and verification of low-level systems code, especially operating-system kernel source-code. The methods are machine-based, that is, the specifications can be processed in various ways, and, most importantly, verification results are machine checkable.

The selected approach for the specification of the Nova interface and for the verification of a selected Nova kernel module is described in much more detail in the deliverable D.6. The detailled formal specification including the pseudocode is available as deliverable D.12 and the verification results are reported in detail in D.13. Below we giva a summary on these results.

RU and TUD: Contrary to the original project plan and to compensate for problems that occurred in the first project phase, a TUD researcher participated in this verification workpackage. Nonetheless we report both partners' activities here.

Specification

Traditionally, a formal specification of a kernel interface is only started after this interface has stabilized. However, Nova was developed in parallel to our specification effort. We therefore had to develop procedures to quickly and precisely communicate changes in the interface between the kernel developers and the formal methods people. The approach that has been pioneerd in this project turned out to be a brilliant compromise that both formal methods people and kernel developers are comfortable with.

The approach we evaluated was to ask the kernel developers to specify the system call behavior by writing pseudo code in an informally defined language and to formalize this description in pseudo code only using simple set theory. Thus, changes to the interface could quickly and precisely be communicated to the formal methods people by adjusting the pseudo code and the formal specification could be proof read by the kernel developers without having to learn the language of modern verification tools.

Similar to the pseudo code, we used an UML like state diagram in preparation for the formalization of the kernel state space. In this diagram all live (i.e., currently active) kernel objects and their relations are shown. The associations in this diagram appear in the pseudo code as pointers and consequently dereferencing such a pointer means to follow the association.

We formalized both the UML diagram and the pseudo code using the Nova interface base specification (Nibs). Nibs is only based on simple set theory and defines the semantics of the language constructs in terms of their effect on the kernel state which is the collection of all live kernel objects. In addition to the common language constructs such as assignment, pointer dereference and if-then-else, Nibs allows the pseudo code developer to use common set and list operations directly and defines in the background a precise formal semantics for these operations. Consequently, although the operations are formally defined, the actual maths is not necessarily visible to the pseudo code developer.

A notable feature of Nibs is the semantics of the block operation. The behavior of most system calls are defined as a single, atomic transition in the kernel state, however, blocking system calls such as IPC reveal an intermediate state that can be modified by concurrently executing threads. The semantics of block automatically breaks the pseudo code of an otherwise atomic transition into multiple transitions (e.g., one from the beginning to the blocked operation and a second after the blocking condition is fulfilled).

Verification

The approach selected for the verification of a Nova kernel module is described in D.6. More detailed results can be found in the respective publication at the following conferences: Nluug [13], Workshop on C/C++ Verification [12] and SSV'08 [14]. Here, we sketch briefly the involved components and our achievements.

To formally verify a piece of C++ kernel code we need a semantics of C++ that is suitable to detect the most common programming errors. In addition we need a precise model of the underlying hardware on which the kernel executes as most of the features exposed by the hardware (such as page tables to establish virtual memory, interrupts, device registers possibly causing side effects) are in fact used by the kernel code. To assert correct behavior of the kernel code we must show that these features cause no unintended effects or insecurities.

We therefore developed a semantics compiler based on Olmar [11], an Ocaml backend for the Elsa C++ parser. The purpose of this compiler is to translate the C++ source code into its semantics written in higher order logic in the PVS theorem prover. The C++ semantics we developed exploits the underlying hardware model for accessing reading data from and for writing data to memory. The C++ semantics defines an underspecified behavior for the subset of all C++ language constructs used in the kernel as it is described in the international C++ standard [1]. The consequent use of underspecification in the C++ semantics [13] as well as in the hardware model serves two purposes. First it allows a concrete, possibly implementation defined and thus compiler specific behavior to be defined later and only for the part of the verification where such a behavior is required. Secondly, it reveals a large class of bugs by leading to proof obligations that can not further be simplified with the specified semantics. An example for the second case is the consequent use of the underspecified **mod** function for modulo arithmetic (see for example D.13 Section 4.7.6). This function is defined to return the identical value of a computation only for those values that are in the representable range of the corresponding C++ data type. Thus it allows for the detection of unintended integer overflows and of imprecise intermediary results in a floating point operation. If however at a certain point in the program, the kernel developer relied on the integer overflow behavior as it is defined in the C++ standard, this behavior can be included into the formal verification by the means of an additional axiom for **mod**.

The hardware model [14] describes a hierarchy of different memory models. In each level of this hierarchy further functionality can be added. For example one such layer defines virtual memory and the corresponding mapping to physical memory that the processor establishes from the information in the page tables. Another layer defines the behavior of a single device. The complexity of this model is required to precisely capture the behavior of the underlying hardware, however it is far from practical to directly verify code against such a complex model. We have therefore developed a property called **plain_memory** [13, 14], which we have proven for each layer of the hardware model. This property states that under certain preconditions and in certain address ranges, the memory behaves as expected (i.e., the last written value is returned when reading and both reading and writing cause no unexpected side effects to other memory in the region). In a source-code verification the established plain memory can now be used instead of the more complex layered hardware model, provided the source code executes only within the above address range.

We exemplified such a verification for a small piece of C++ code with pointers and pointer arithmetic. This code runs in virtual memory and thus may access variables in disjoint regions of virtual memory which are however mapped to the same locations in physical memory. We call these virtual-memory aliases. Furthermore we show in this example verification how a simple memory mapped device with access and alignment restrictions, reserved bits in the device registers and side effects can be integrated in our verification approach and how the complexity of the virtual to physical address translation as well as for the device was hidden by the plain memory abstraction.

For more details on this layered verification approach please refer to Section 5 in [14] and Section 4.8 in D.13.

WP5: Application Scenario

The objectives of this workpackage were to design and implement a real-life scenario to illustrate the feasibility of ROBIN and its usage towards potential end users. A further objective was the communication of end user concerns and requirements to the ROBIN developers.

secunet: As a company with excellent contacts to potential end users of the ROBIN architecture in form of its customers, secunet performed a requirements analysis for usage scenarios (D.7) and implemented a research demonstrator (D.14) showing the feasibility of ROBIN by means of one such scenario.

Usage Scenario Requirement Analysis
Secunet performed an analysis of several scenarios to derive the requirements on the ROBIN architecture. To highlight two important results of this analysis, the possibility to split applications into a large security-insensitive part and into a small security-sensitive part and to run the latter on a minimal TCB was identified as an important property of the ROBIN architecture. The security-sensitive parts have been identified in several scenarios. These scenarios include virtual-private-network gateways, secure document editing and secure email processing. The second important result is that fall-back mode operation, with the exception of using this mode to initiate a recovery of the normal mode, is hard to motivate towards potential end-users. Among the major concerns expressed were issues related to the usability of the system under reduced functionality and user-interface issues that allow quick adaptation to the changed look-and-feel of an application in fall-back mode. On the other hand, we identified several mobile scenarios in which fall-back mode functionality is easily accepted. This can for example be seen in the embedded demonstrator showing that in fall-back mode emergency calls can be placed despite of a successful attack to the mobile phone's legacy OS.

Real Life Scenario
One of these scenarios — the secure document editing — has been selected for the demonstrator of a real-life scenario and was subsequently designed and implemented.

This scenario is comprised of a session management component which controls the operation of isolated instances of a legacy operating system. Fast startup of editing instances is enabled by L4Clone, a component which allows bootstrapping new legacy OS instances from a frozen system image. Secure gateways are responsible for the transfer of documents from the document repository to the editing sessions. User-friendly and secure interaction with multiple isolated sessions is enabled by an extension to Nitpicker, a secure display multiplexer.

The deliverables on the requirement analysis and on the real-life scenario are not intended for publication.

TUD: TUD's involvement in this workpackage was to provide assistance, help and support for the microkernel / microhypervisor and trusted-computing-base toolkit on which the real-life scenario was developed.

WP6: Project Management

This workpackage concentrates all project management related efforts. These include the organization of project meetings and the filing of the present and further intermediate and final reports as required by the European Commission. A further objective of this project-management workpackage was the organization and assembly of an end-user advisory board.

In ROBIN we had 3 project meetings to which we invited the project officer and at which we presented our plans and achieved results. These meetings were a kick-off meeting, a meeting at the first milestone (approx. in the middle of the project) and a meeting at the end of the project. In addition we organized two additional meetings to which only the project members attended and with the objective to communicate intermediate results also to those partners that were not directly involved in the work and to plan the work in the respective next half-year period.

Accompanying each of these meetings we filed activity and project management reports describing the work performed in the respective elapsed period.

Advisory Board
We have contacted several potential end-users of the ROBIN architecture and received valuable feedback, however, due to the strict timing constraints of these highly skilled people we were unable to assemble these end users in an advisory-board meeting.

To compensate for this assembly we visited potential end users of the ROBIN architecture individually and discussed with them our technology and achievements. We would like to express our thanks for their valuable feedback.

Our dissemination activities of the ROBIN results does not only addresses the potential end users of this technology and the sciencific audience at research conferences but also non-scientific people. For example, on February 19th, 2008 Peter van Rossum gave an interview on Dutch national radio about the Robin project. The interview, in the radio show called "Radio Onlinea", is available at [15].

Additional Results

To achieve the deliverables in a project, experiments have to be focused towards the objectives, however, a broad applicability of the ROBIN architecture is a key success factor. We therefore tried not to rule out applications of ROBIN in a real-time system setting as well as in high-secure settings that both exceed the requirements of those scenarios that we envisaged in the scope of this project. Consequently we performed the following experiments, the results of which we published at several international conferences.

The emergence of commodity multicores and the desire for real-time guarantees not only affects kernel and systems research, algorithms in user applications have to be re-evaluated by the research community under the new preconditions set by those systems. Looking for a typical application that is demanding even for the fastest single core machines, we found high definition video decoding to be an interesting workload. It allows exploration of both function parallelism by pipelining the video decoding steps and data parallelism by spatially distributing a frame's pixels to cores. In addition, it forms a workload with natural real-time requirements. To examine the scalability problems of video decoding, we parallelized a H.264 video decoder and looked into potential optimizations of the video stream to utilize the cores more efficiently. This work has been presented as a full paper on EMSOFT 2007 [9]. To achieve more accurate real-time guarantees, a decoder model was developed to generate very accurate decoding time predictions for use by a real-time scheduler. This work has been presented as a full paper on RTSS 2006 [10].

For use in a high-security setting, absence of illegal information flows must be asserted. We investigated a modification to the real-time scheduler of Nova that is capable of avoiding illegal timing channels [18] and that achieves timely isolation as required by the common application of time triggered systems. We achieve this while preserving the system's real-time properties. Furthermore, we investigated how static program analysis techniques can be applied to assert the confidentiality of security sensitive applications that may communicate through shared memory [16].

1.1.4. Impact on the Security of Citizens and on future Industrial and Academic Research

Possibly the most significant impact of a future wide deployment of the ROBIN architecture is an increased trust in and an increased reliability of information and communication infrastructure. This is due to the continuing effort to reduce the per application trusted-computing base to what is absolutely required. As can be seen in the demonstrators originating from this project, the size of the components that remain in the TCB has again reached an order of magnitude that allows their fairly complete understanding and that makes it even possible to tackle their formal verification.

In the following section we highlight more concretely where we see an immediate impact of ROBIN on future industrial and academic research.

As already pointed out, we limited the experiments that were not in straight line to achieving the promised deliverables. As such, these experiments are the natural next steps in terms of academic and industrial research. These experiments include: scalability of the ROBIN architecture to the upcoming, possibly heterogeneous multi-core architectures, real-time, timely-isolated and high-secure systems, embedded systems with much tighter memory limits.

The research challenges we can conclude from the investigation of real-life scenarios include graphical user interfaces for emergency mode operation and other methods to increase the intuitive use of a mobile device in an emergency situation.

Despite the success of formally verifying a selected kernel module, we have to focus on techniques that make formal verification scale to larger quantities of code. In particular, the Bastei components form an obvious next verification target as does a complete verification of a microhypervisor. Furthermore, methods have to be developed to further automate the verification respectively to make previous time-consuming verification efforts reusable for future versions.

1.2. Conclusions of Activity Report

We have presented the activities performed and the results achieved in the ROBIN project as well as how these enhance the state-of-the-art. The results of all academic partners will be released under an open source license within the next few month.

Please contact the academic and industrial partners for further information and to discuss additional licensing options.

A. Final Plan for Using the Knowledge and for the Classification of Results

A.1. Final Plan for Using the Knowledge

- exploitable results
- impact of these
- the way they will be used
- IPR protection of the results

To do: This is the Annual Report Text; Fix it

To do: Note, the Comission may review the actual implementation of this plan (Article II.34 grant agreement)

In the following we describe our plan for using the results of this project. The Consortium Agreement of this project formalizes that at the end of this project, software owned by the academic partners (TUD - D.2, D.4, D.9, D.10; RU - D.12, D.13) will be published under an open source license following the general principles of the GPL (version 2). The software of the remaining deliverables are owned by the respective industrial partners (ST - D.5, D.11; secunet - D.7, D.14) and will not be published. Please contact the academic and industrial partners for additional licensing options.

A.1.1. Publishable results

This project achieved the following publishable results:

Nova (TUD): An implementation of the Nova microhypervisor on the Intel VT platform. This research prototype allows for future research activities on microhypervisors and microhypervisor-based systems and could be the basis for future productizing efforts. Specific research directions include microhypervisor-based multicore architectures, real-time systems and high-security systems.

Further stabilization and testing is required for the Nova implementation which justifies a delay of the open source release of this prototype. We expect a release of Nova and of the accompanying specification in Q4 2008.

Bastei (TUD): Implementation of the Bastei - TCB Construction Kit. This research prototype allows for future research activities in the area of complexity reduction of security sensitive applications and in the area of constructing operating systems featuring a minimal-trusted computing base. Productizing efforts have been started in a recently founded spin-off company.

Formal Specification of the Nova Interface (RU): This formal specification enables formal reasoning of the properties of the Nova Interface including the adherence of a particular implementation to this interface. We are confident that the applied methods can be generalized to formally specify other microkernel- and microhypervisor interfaces.

Formal Verification of a selected Kernel Module (RU): This formal verification of a selected kernel module illustrates an approach for the source level verification of microhypervisors and gives an estimate on the costs involved. We plan to extend this verification approach and to verify a significant part of the microhypervisor in future research projects.

The sources of STMicroelectronics' minimal TCB —an implementation of ROBIN on an embedded platform— and the demonstrator of a real-life application scenario (secunet) are not foreseen for release after this project.

To obtain information on licensing possibilities for the deliverables (including non GPL licenses of the published deliverables) please contact the respective partners:

Contact Information

To contact the ROBIN Project members please send an email to: `contact@robin.tudos.org`

A.1.2. Exploitation and Use Potential

The following lists the results we generated during the ROBIN project and their exploitation and use potential.

Results Generated	Owner	Sector(s) of Application	Exploitation / use potential or actual measures
Nova Architecture Whitepaper	TUD	VM-Products, OS-Research	determine suitability for customer VM solutions future research on microhypervisors
Nova Prototypical Implementation	TUD	OS-Research	future research on microhypervisors productization
Bastei Architecture Whitepaper	TUD	OS-Research	future research on microhypervisor-based systems
Bastei Implementation	TUD	OS-Research	future research on microhypervisor-based systems
Note on Methods for Verification	RU	Verification, Research	applicability to own verification approach future research on microhypervisor verification
Nova Specification	RU	Verification, Research	verification at top-level specification level future research on microhypervisor verification
Nova Verification of Kernel Module	RU	Verification, Research	verification of entire kernel future research on microhypervisor verification
Split of Embedded Application	ST	Embedded Systems	suitability of Robin platform and the split application approach for embedded solutions
ROBIN on Embedded	ST	Embedded Systems	future research on ROBIN embedded productization
Requirement Analysis	secunet	Security Systems	determine suitability of Robin platform for security system solutions
Real Life Scenario	secunet	Security Systems	determine suitability of Robin platform for security productization

The beneficiaries have partially started using the results of this project in their future research and product activities. TUD and RU already integrated the achieved results into their advanced operating-system courses and in their student projects (e.g. master theses). Most of the results of the academic partners have been licensed under the terms of an open source license following the general principles of the GPL v2. An open-source release of the Nova implementation is foreseen in Q4 2008.

A productization was not targeted within the scope of this preparatory research activity.

A.2. Final Plan for Classification of the Result

The information involved in this Activity is not classified.

B. Milestone and Deliverable List

Milestones List

Milestone no.	Milestone name	Workpackage no.	Date due	Actual/Forecast delivery date	Lead beneficiary
1	M.1	all	31/01/2007	31/01/2007	TUD (coordinator)
2	M.2	all	31/04/2008	31/04/3008	TUD

Deliverables List

Del. no.	Deliverable name	WP no.	Nature[14]	Dissemin. level[15]	Date due	Delivery date	Estimated indicative person-months	Used indicative person-months	Lead beneficiary
D.1	Microhypervisor Architecture Whitepaper	WP.1	R	CO	M.1	M.1	2	2	TUD
D.2	Microhypervisor Interface Specification	WP.1	R	RE	M.1, M.2	M.1, M.2	6	3.28 +	TUD
D.3	ROBIN Architecture Whitepaper	WP.2	R	PU	M.1	M.1	2	2	TUD
D.4	Informal Specification of Key TCB Components	WP.2	R	PU	M.1, M.2	M.1, M.2	6	3.5 +	TUD
D.5	Split of Embedded Application	WP.3	P	CO	M.1	M.1	6	7	ST
D.6	Note on Selected Verification Approach	WP.4	R	PU	M.1	M.1	6	6	RU
D.7	Usage Scenario Requirements Analysis	WP.5	R	CO	M.1	M.1	9	7	secunet
D.8	Project Reports	WP.6	R	CO	M.1, M.2	M.1, M.2	13	4+	TUD
D.9	Implementation of Microhypervisor on VT	WP.1	P	PU *)	M.2	M.2	45	20+	TUD
D.10	Implementation of ROBIN TCB Kit	WP.2	P	PU	M.2	M.2	52	28.25+	TUD
D.11	ROBIN on embedded platform	WP.3	P	CO	M.2	M.2	12	5.5+	ST
D.12	Machine Checked Specification	WP.4	R	PU	M.2	M.2	20	1+	RU
D.13	Machine Checked Verification	WP.4	R	PU	M.2	M.2	33	14+	RU
D.14	Real-Life Scenario	WP.5	P	CO	M.2	M.2	15	2+	secunet

***) This demonstrator will be released in 2008 under an open source license following the general principles of the GPL v2.**

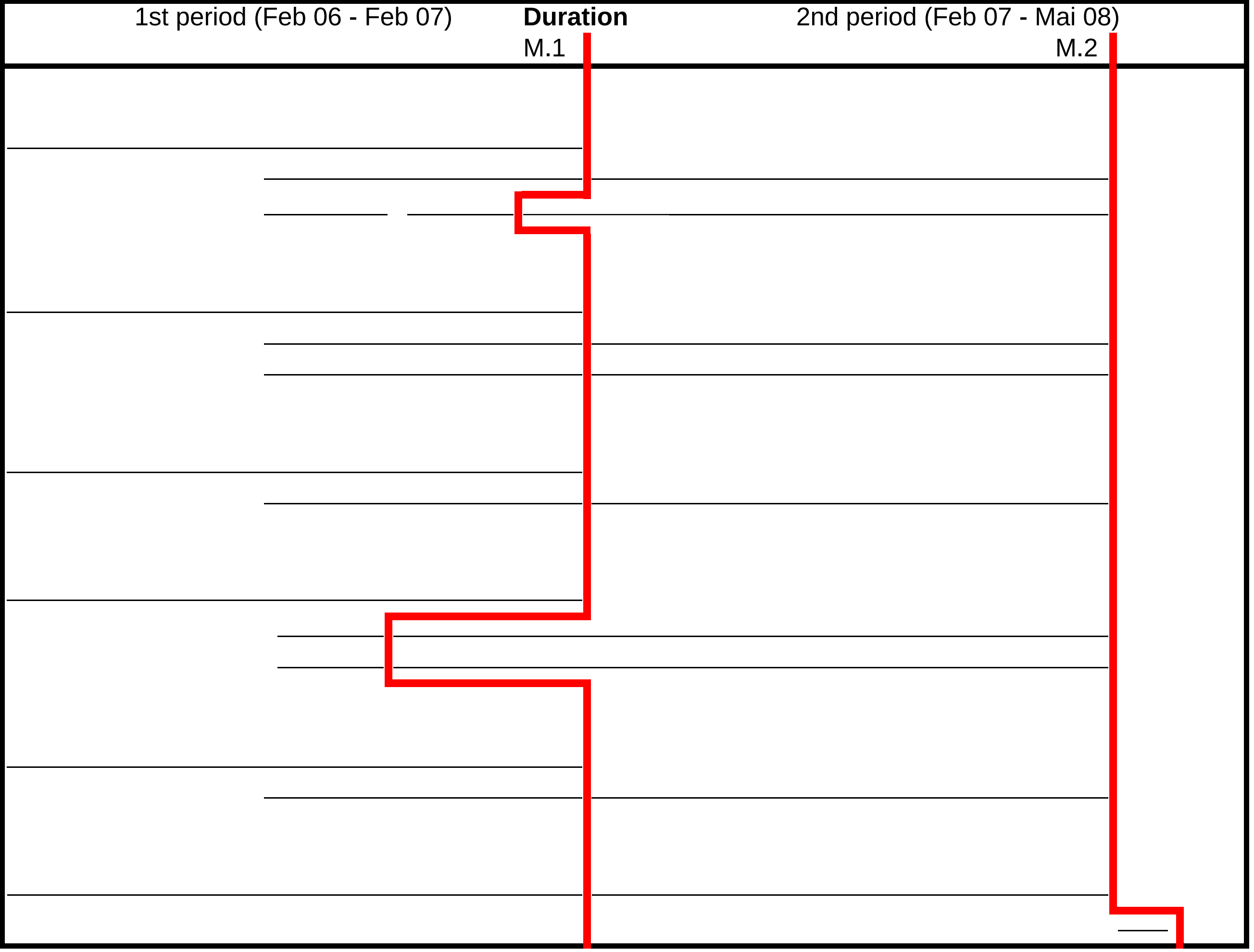
[14] Please indicate the nature of the deliverable using one of the following codes:
- **R** = Report
- **P** = Prototype
- **D** = Demonstrator
- **O** = Other

[15] Please indicate the dissemination level using one of the following codes:
- **PU** = Public
- **PP** = Restricted to other programme participants (including the Commission Services).
- **RE** = Restricted to a group specified by the consortium (including the Commission Services).
- **CO** = Confidential, only for members of the consortium (including the Commission Services).
- **CL restricted** = Classified with the mention of the classification level restricted
- **CL confidential** = Classified with the mention of the classification level confidential
- **CL secret** = Classified with the mention of the classification level secret
- **CL top secret** = Classified with the mention of the classification level top secret

C. Activity Barchart and Status

Activity Barchar and Status

Acronym:	Robin
Contract No:	SEC5-PR-104600



Workpackage 1:	
	Microhypervisor
D.1	Architecture Whitepaper
D.2	Interface Specification
D.9	Implementation on VT
Workpackage 2:	
	TCB Construction Kit
D.3	Architecture Whitepaper
D.4	Informal Specification
D.10	TCB Kit Implementation
Workpackage 3:	
	Embedded Platform
D.5	Split of Embedded Application
D.11	ROBIN on Embedded Platform
Workpackage 4:	
	Kernel Specification and Verification
D.6	Note on Verification Approach
D.12	Machine-Checked Specification
D.13	Machine-Checked Verification
Workpackage 5:	
	Application Scenario
D.7	Requirements Analysis
D.14	Real-Life Scenario
Workpackage 6:	
	Project Management
D.8	Project Reports
	Financial Report

Bibliography

[1] *Programming languages - C++ Internatilnal Standard*, iso/iec 14882 edition, Okt 2003.

[2] ARM. Trustzone - Security Extensions to the ARM Architecture. http://www.arm.com/products/esd/trustzone_home.html.

[3] A. Chou, J. Yang, B. Chelf, S. Hallem, and D. Engler. An empirical study of operating system errors. In *18th ACM Symposium on Operating Systems Principles (SOSP)*, pages 73–88, 2001.

[4] N. Feske. A Case Study on the Cost and Benefit of Dynamic RPC Marshalling for Low-Level System Components. *SIGOPS OSR Special Issue on Secure Small-Kernel Systems*, 2007.

[5] B. Jacobs and E. Poll. Java Program Verification at Nijmegen: Developments and Perspective. In *Software Security - Theories and Systems (ISSS'03*, Springer LNCS 3233, pages 134–153, 2004.

[6] W. Knight. Prank leads to fears about mobile security. http://news.zdnet.co.uk/internet/0,1000000097,2080733,00.htm, Aug 2000.

[7] EU IST Project. Mobius. http://mobius.inria.fr, 2001 - 2003.

[8] EU IST Project. VerifiCard. http://www.cs.ru.nl/verificard, 2001 - 2003.

[9] Michael Roitzsch. Slice-Balancing H.264 Video Encoding for Improved Scalability of Multicore Decoding. In *Proceedings of the 7th International Conference on Embedded Sofware (EMSOFT 07)*, Salzburg, Austria, October 2007.

[10] Michael Roitzsch and Martin Pohlack. Principles for the Prediction of Video Decoding Times applied to MPEG-1/2 and MPEG-4 Part 2 Video. In *Proceedings of the 27th IEEE Real-Time Systems Symposium (RTSS 06)*, Rio de Janeiro, Brazil, December 2006. IEEE.

[11] H. Tews. OLMAR C++ Parser. available for download at: `http://www.cs.ru.nl/ tews/software.html`.

[12] H. Tews. Formal Methods in the Robin project: Specification and Verification of the Nova microhypervisor. In *Workshop on C/C++ Verification*, Okford, UK), month = July, note = available at: `http://www.cs.ru.nl/ tews/science.html`,, 2007.

[13] H. Tews. Micro Hypervisor Verification: Possible Approaches and Relevant Properties. In *NLUUG Voorjaarsconferentie*, Ede, NL), month = May, note = available at: `http://www.cs.ru.nl/ tews/science.html`,, 2007.

[14] H. Tews, T. Weber, and M. Völp. A Formal Model of Memory Pecularities for the Verification of Low-Level Operating-System Code. In *Nicta 3rd International Workshop on Systems Software Verification (SSV08)*, Sydney, Australia, Feb 2008.

[15] P. van Rossum. Interview about the robin project in radio onlinea. Interview on Dutch national radio, Feb 2008. available at: http://cgi.omroep.nl/cgi-bin/streams?/radio1/tros/radioonline/20080219-20.wma?start=29:40.

[16] M. Völp. Statically checking confidentiality of shared-memory programs with Dynamic Labels. In *The Third International Conference on Availability, Security and Reliability (ARES 2008)*, pages 268 – 275, Barcelona, Spain, March 2008.

[17] M. Völp, S. Courcambeck, R. Dorn, H. Härtig, and H. Tews. *D.8 Annual Report Milestone M.1 - Executive Summary*, Feb 2007. available at: http://robin.tudos.org/publications.

[18] M. Völp, C. J. Hamann, and H. Härtig. Avoiding Timing Channels in Fixed-Priority Schedules. In *ACM Symposium on Information, Computer and Communication Security (ASIACCS '08)*, Tokyo, Japan, March 2008.