NICTA

Trustworthy Embedded Systems

ERTOS-2 Project Plan 2009–2013

ertos@nicta.com.au

July 2009

Abstract

This report presents the research agenda for NICTA's ERTOS team for the four years from October 2009. It builds on the successful seL4, L4.verified and CAmkES projects to take an ambitious next step towards truly trustworthy real-world systems. Specifically, it aims to provide strong dependability guarantees for systems built from trusted and untrusted (including legacy) components, and comprising millions of lines of code.

1   Summary

The Problem. Complex embedded systems consist of millions of lines of code and components of varying quality, but have high-assurance requirements for the whole system.

The Approach. Leveraging our unique position with a verified kernel and a strong FM/OS team, we will develop a platform and method that integrate with current development processes to achieve whole-system guarantees with minimal formal methods involvement for the developer.

The Outcome. We will be able to build systems with millions of lines of code affordably, with strong, formal whole-system guarantees.

The Outputs. We will create tools for implementing high assurance systems, formal models and methods for analysing them, and a set of critical trusted OS components to build on.

Path to Impact. Partnering with OKL and defence we will focus initially on the defence vertical market while exploring further market possibilities during the course of the project. Candidates are avionics, bio-medical, SCADA and automotive sectors.

2   Technology and Science

2.1   Research and potential impact

The ERTOS-2 project aims to solve the problem of building complex embedded software systems with millions of lines of code, while at the same time providing formal guarantees of critical safety and security properties. Formal verification is one of the main differentiating factors of this project, but the research challenges involved to make such systems practical and feasible include far more. In particular, we propose to attack cross-cutting OS research concerns that are vital in the embedded systems application area. These include multi-core processors, energy consumption, providing temporal isolation, trusted OS components, and microkernel-based system architectures.

Industry and consumers increasingly rely on modern embedded systems to perform critical tasks such as controlling medical devices, mobile phones, industrial plants, Internet infrastructure, automobiles, and aeroplanes. These systems are a complex combination of hardware and software, with the software typically accounting for a significant (and frequently dominant) fraction of the overall functionality, complexity and development cost. The software in these systems is huge: the size of the software in a typical 3G smart phone is on the order of 5–7 million lines of code (loc); in support systems for communications satellites 5–10 million loc; and in a high-end car up to 100 million loc. Unfortunately such large, complex, and feature-rich software systems are difficult, expensive, and often impossible to construct such that they can be guaranteed to perform reliably, safely and securely. This ultimately leads to a trade-off between embedded system functionality on the one hand and safety, reliability and security properties on the other.

Software certification standards like the Common Criteria [8] explicitly require the system to be simple for the highest levels of assurance[1] (i.e., EAL7). Otherwise, so it is believed, no reasonable assessment of high assurance of its security or functionality can be made. *Simple* is currently interpreted as few thousands lines of code. Prior to our successful L4.verified project [17,35], which proved the functional correctness of our seL4 microkernel [9], the largest real systems with formally verified functional correctness on the implementation level were on the order of hundreds of lines of code. The L4.verified project pushed this number to roughly 10,000 lines. In the proposed project, we will achieve formal guarantees of specific properties such as security and fault isolation

---

[1]Common Criteria defines 7 evaluation assurance levels (EAL), with EAL1 being the most basic and EAL7 being the most stringent. Formal methods are required from EAL6 on in the latest CC version (3.1)

for systems with millions of lines code. These guarantees will be proven to hold not merely on high-level models, but on the source code implementation level — on the system as it is deployed.

One of the key problems in the current state of the art is that there are no existing verification approaches that both provide strong guarantees and scale to systems with millions of lines of code. Formal verification provides the strongest guarantees. However, even after significantly pushing the state of the art, the largest program whose implementation has been fully formally verified is our seL4 microkernel with only 10,000 loc. Alternative approaches, including process certification, code inspection and testing, rely on informal techniques, thus resulting in lower levels of assurance than formal verification. Furthermore, they are expensive procedures and become prohibitively expensive as software complexity grows. As mentioned above, certification processes explicitly require the software under consideration to be simple, and even then, full certification cost is estimated with about $10,000 per line of code for EAL6, i.e., of the order of $100,000,000 for a program like seL4. EAL7 certifications are so rare that we could not find published cost estimates.

In the application space of defence intelligence, for instance, a common problem that requires highly-certified and -secure software is that of accessing multiple data sources of different classification levels and networks at the same time. Currently, to separate (for instance) highly classified, top-secret Australian data from classified NATO data, two separate networks and access terminals are used. This solution clearly does not scale if the number of networks and data sources grow, leading to an equally large number of computer terminals on the desks of analysts, which leads to inefficient and awkward workflow. The ideal solution would be a terminal that is connected to all networks at the same time and that allows simultaneous access to these levels, for instance in separate windows. Indeed, there are such commercial solutions available, for instance, from Green Hills Software (GHS), Trusted Computer Solutions (TCS), or Raytheon. These systems commonly build on large operating system bases like Windows NT, SELinux, or Trusted Solaris. None of these are certifiable to be used in potentially hostile environments [31]. Even GHS's own recent certification to EAL6+ concerns only the Integrity microkernel, not the multi-level secure terminal itself. Nevertheless, the pressure to use these multi-level systems is huge. For instance, TCS's product was recently (mid 2008) approved to be used up to classification level *secret* in the US [21], even though it is based only on a SELinux platform. It is clearly a concern for security agencies that such systems currently cannot provide higher assurance than EAL4.

In the academic space, the idea that separation kernels and microkernels with isolation properties can be used as a secure foundation for such systems has been around since the late 1980s. It is most pronounced in the MILS (multiple independent levels of security) community with John Rushby as one of the main proponents [5]. The main focus of the MILS approach is security, and its basic idea is to use separation to drastically reduce the *trusted computing base* (TCB) of the system, the part that is security critical. While there is a large body of research on the MILS approach, beginning with partition kernels [28] and most recently looking component integration [6], reasoning either remains on high-level models or concerns the separation kernel foundation only. We intend to build on this research, but we will give assurance not just about models and architectures, but instead for the actual system code that is eventually deployed. The difference is similar to what the L4.verified project achieved for the seL4 microkernel, but this time lifted to the whole system, not just the OS kernel. We also propose to attack the problem from the engineering and operating-systems side to make development of these systems practically feasible and not just theoretically achievable.

Our research strategy in microkernels has aimed at exploiting and building on separation from the beginning [12]. Modern microkernels make this idea practical and they can be used to limit the system's trusted computing base to a very small number of components. This reduction of the trusted computing base is not only useful in the security and MILS area, it can also be used for increased safety and reliability [3].

To illustrate this, consider a modern car which features over 100 processors to implement various
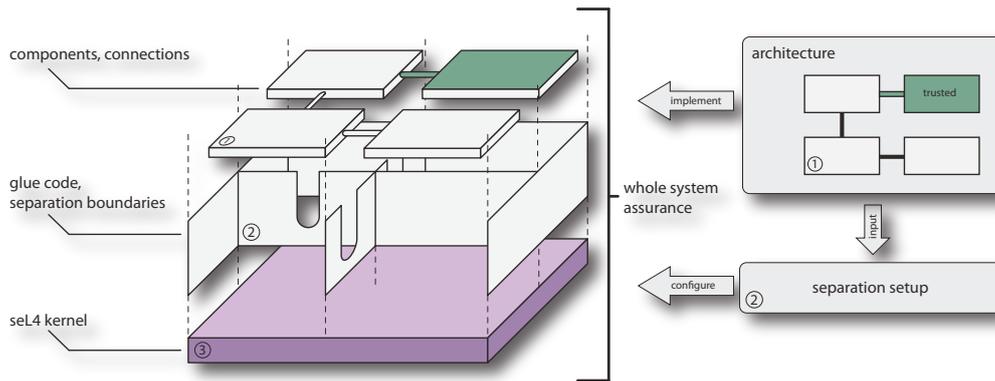
Figure 1: seL4-based system with multiple independent levels of assurance

features of the car, such as multimedia entertainment, car information displays, and braking safety systems. The current trend is towards consolidating functionality on a reduced numbers of processors (for example, Toyota is aiming to reduce the number of processors in a typical car from over 100 to less than four) [23]. Consolidation opens the opportunity for faults in the non-critical multimedia system, to corrupt important displays for the driver, and potentially even affecting safety systems, for example, where displays and braking both use shared speed sensors. In such a system, the non-critical multimedia system is part of the trusted computing base as it may cause the whole system to fail. A similar scenario involving multimedia systems and safety critical components has recently lead the FAA to release a "special conditions" document [2] regarding the new Boeing 787.

By isolating critical components from non-critical components, system developers need verify only the truly critical parts of the system, rather than the entirety of the system. This reduces overall effort, while still providing the required guarantees regarding levels of safety and security.

Our solution will enable embedded systems vendors to focus their efforts (and budgets) on critical system functionality, allowing non-critical and less trusted software to be incorporated from third-parties, all the while being able to trust (and prove) that the underlying platform ensures strong isolation of this less trusted software. In this way, embedded-systems vendors can deliver systems with complex feature sets possessing differing levels of trust, or systems consolidating previously separate systems, and still have ultimate assurance of the functionality and validity of the critical components in the system.

The ERTOS-2 project will develop the methodology to build such systems, will provide smart security architectures that achieve the desired reduction of the trusted computing base, will develop the necessary infrastructure to automatically analyse and implement these architectures in code, and will provide verification tools and methods to show that the reduced number of trusted components are indeed trustworthy. The result leverages isolation guarantees for the construction of systems that are composed of both highly-trustworthy critical software and non-critical untrusted software at the same time. Our approach builds on a formally verified, trustworthy microkernel-based operating system, tools for construction and analysis of component-based systems and trustworthy components, and the unprecedented enforcement of functional and temporal isolation boundaries between components, without increasing the cost of the software and its development.

Our approach consists of three key parts, and is illustrated in Figure 1. We will demonstrate the approach by building a multilevel-secure terminal as described above, but we emphasize that the resulting methodology and tool set are more generally applicable. In the case of the secure terminal solution, the approach begins with the specification of required system security properties, here an information flow property. In the general case, this may also be a safety or reliability property such as integrity or fault isolation. Given this specification, the system's software architecture (1)

is designed such that it achieves the desired security goal with a small trusted computing base. In particular, the architecture description must specify the components involved, must specify the connections between the components, and must identify the trusted components as well as any isolation and communication restrictions. The architecture is designed using existing design methodologies, allowing existing architecture analysis methods to be used for these high assurance systems.

In the architecture, trusted components are those that could circumvent the reliability or security properties of the system. In order to maintain the required security or reliability properties, trusted components are expected to behave in a particular way, and these behavioural requirements are specified as part of the architecture.

The architecture description is then validated against the system policy to ensure that it does in fact provide the required properties. For specific sets of security and safety properties such as label-based security policies we will provide tools that can automate this analysis and produce formal proofs.

The next step is to provide implementations of the components specified by the architecture. These components can be OS components (drivers, resource management, etc.) that form part of our OS platform, middleware components (such as naming and communication servers), and application specific components. The components are either implemented from scratch, or chosen from a library of pre-existing reusable components.

For assuring trusted components there are many existing approaches and much research has investigated component creation. We do not mandate any particular approach, however, the approach taken should provide sufficient confidence that the components fulfil their requirements. We will then provide a formal framework into which the given individual assurance can be integrated. One such approach is formal code proof as performed in the L4.verified project. Another such approach is the synthesis of component implementations from formal specifications. In particular we are developing approaches for automatic synthesis of device driver components. This approach is less resource-intensive than manual proof and can produce a similar level of assurance.

Finally, given the architecture description and component implementations, we will generate the code required to combine the component implementations into a running system. This includes generated glue and communication code as well as initialisation and bootstrap code (which is responsible for loading and initialising the required components, creating the appropriate partitions (2), and configuring communication channels as specified by the architecture). This step will be performed automatically by a tool (evolved from our CAmkES project [18] that developed a lightweight component framework for embedded systems) that will also automatically integrate the proofs of architecture correctness, component correctness, partition configuration and kernel correctness into one overall statement on the whole system.

Summarising with reference to Figure 1, this approach results in components (the top layer in the diagram, labelled 1) assured to the individual levels that the overall system requires, and a system architecture shown to enforce the component isolation and communications requirements (the walls in the diagram, labelled 2). These are composed such that individual component assurance combines into strong system-level assurance, and are supported by a platform that provides the system's execution environment and isolation guarantees (the seL4 kernel, labelled 3). The important part is that the separation walls come with a strong foundation that enables us to ignore non-critical components for assurance purposes.

Our approach differs from, and is a significant improvement over, existing approaches in two key ways. Firstly, our previous work has provided a foundation in the form of the seL4 microkernel (in the seL4 project), which is unique in that it has been formally proven to correctly provide separation and to constrain communication and resource sharing (in the L4.verified project), together with a framework for component architectures for embedded systems (in the CAmkES project). Secondly,

given a system architecture, we expect to develop tools that instantiate the architecture with similar formal guarantees of correctness. **Our vision is that both the seL4 foundation and the architecture instantiation are built and proven correct with mathematical rigour from the design down to the code level.** We also expect that our approach will be readily re-deployable in alternative applications (by providing a different architecture and combining this with alternative components) with minimal input from formal methods practitioners, while retaining the formal guarantees and foundations that the approach is based upon.

As mentioned above, the basic idea of the approach is similar to MILS systems in the security area, but we propose to carry through the MILS vision for entire realistic systems down to the code level. Significant code-level proofs are currently conducted in the Verisoft project [13, 29] and its successor Verisoft XT [36]. The former did involve formal guarantees for a whole system (more than the OS level), but to achieve this goal, the system was simplified significantly and with overall around ten thousand lines of code several orders of magnitudes smaller than what we aim at. The latter, Verisoft XT, aims at the Microsoft Hypervisor and is still in its early phases. We are not aware of plans to lift the results to whole systems. On the commercial side, GHS comes closest with their EAL6+ certification of the Integrity kernel and products in the multi-level security space. These products do not achieve the levels of assurance we are proposing, though, and not even the certified kernel is close to what we already have achieved. The same is true, only more so, for other commercial multi-level security vendors like Trusted Computer Systems in the defence area or Industrial Defender in the SCADA area.

We believe we can achieve the goal we have set ourselves, because the proposed research challenges strongly build on our existing experience and tool sets for code-level proofs on the microkernel. We have a proven track record for research and commercialisation in the intersection between formal methods and operating systems research and we are the only group in the world with critical mass and long experience in this intersection. The security sector is a newer application direction for our larger team, but with Ron van der Meyden and Kevin Elphinstone we have two experts in the field as part of the project. We also have strong interest from three commercial partners (defence, Galois, and Intel) in our research agenda.

Success of the ERTOS-2 project will transform the way high-assurance systems are built. Applications in national security and life-critical devices will be able to be assured to levels that are desirable but presently unfeasible. Furthermore, medium to high assurance will become possible where presently low-level assurance or no assurance at all is being used (at a risk of significant financial loss or compromised privacy). This applies to such everyday devices as mobile phones, chip cards, point-of-sale devices, automobiles, wearable medical devices or even enterprise IT systems, mainstream web services and servers. The potential impact of the technology is huge, and goes well beyond the possible direct commercialisation outcomes. Our long-term vision is a world where "the system crashed" is the equivalent of today's "the dog ate my homework".

The project attempts to take bold steps into new directions. The first year will therefore see an emphasis on gaining new experience and recruiting researchers with complementary know-how (we have two current applicants and will be looking for a third). At the end of the first year we will re-assess the project plan, including decisions on which work packages to continue, to stop, or to emphasise, and develop more specific milestone plans for the surviving work packages. Ideally all work packages will continue with the proposed resources. We have identified the packages with the highest risk and designed the project such that they are not on the critical path to overall success.

The specific outputs of the first year will be an initial prototype design and partial implementation of a multilevel secure terminal, demonstrating the approach. This includes simplified prototypes of the formalisation frameworks on each level (architecture, booter/glue code, trusted components), an integrated build system and configuration management tool for seL4-based systems on the engineering side, and case studies in assembly verification, information flow properties of seL4,

and the whole-system framework. The main outcome after the first year will be a solid basis for deciding how to proceed into the next stage.

At project mid-term (after Year 2), clear progress should be visible for each of the framework packages (formal and engineering) such that they can be applied to the multi-level secure terminal design in the following two years, to be refined into practical methods for providing affordable whole-system guarantees.

The outcome of the project after year four will be a way of reasoning about, architecting and developing large-scale embedded systems that are formally verified to satisfy specific security requirements. The project outputs are: a method for deriving architectures with small trusted computing base for specific security policies; tools for automatically instantiating such architectures with an automatic proof of correctness of the instantiation; tools for synthesising trusted device drivers; tools and methods for formally verifying trusted components on the C and assembly level; methods for using existing software architecture approaches for architecture modelling and analysis; a power-management system for microkernel-based embedded software; and a multicore-enabled seL4 microkernel providing spatial and temporal isolation.

The application-level output of the project will be a detailed design of a multilevel-secure terminal solution with reduced trusted computing base and a prototype implementation of such a terminal. The design will be formally modelled and analysed, with an overall system theorem over its chosen security property from the design to the code level. The prototype will demonstrate that our tools and methods are feasible and practical for systems with over a million lines of code.

2.2 Research challenges

In Figure 2 we summarise the structure of the project. The main research challenges are roughly aligned with the abstraction levels described in the previous section: architecture, trusted components, architecture instantiation, and the microkernel. The layers are connected and integrated in the final whole-system theorem that provides the unifying framework. We address cross cutting concerns like concurrency and energy on each level, but summarise their description on the kernel level where they are most pronounced. In the following we describe the major research challenges in each of these areas.

Security Architecture. The main challenge in providing high assurances to more than one million lines of software is to be able to concentrate the rigorous formal verification effort to a minimal trusted computing base (TCB) associated with the properties of interest, and deal with individual properties being dependent on differing subsets of the overall system. Providing guarantees for a property's trusted subset needs to imply an overall assurance of the property for the whole system.

We aim to develop classes of architectures suitable for microkernel-based systems such as seL4. The goal of reducing the trusted subset of the system is in tension with both the high-level functionality of the application domain, and the need to manage low-level hardware resources which may need to be shared. The application domain pulls a system's architectural boundaries towards the application's security and safety boundaries, while the hardware platform drives architectural boundaries towards device and platform boundaries. The two forces on the architecture are in conflict, unless hardware reflects the application safety and security boundaries, which is rarely the case. The project aims to explore this conflict, and to develop architectures that are particularly suitable for reducing the trusted computing base of such systems, are readily (re-)deployable to different application domains, and are amenable to formalisation.

We will design and formally model such architectures on a high, abstract level, by giving a formal semantics to a simple component/communication channel architecture that builds on strong isolation
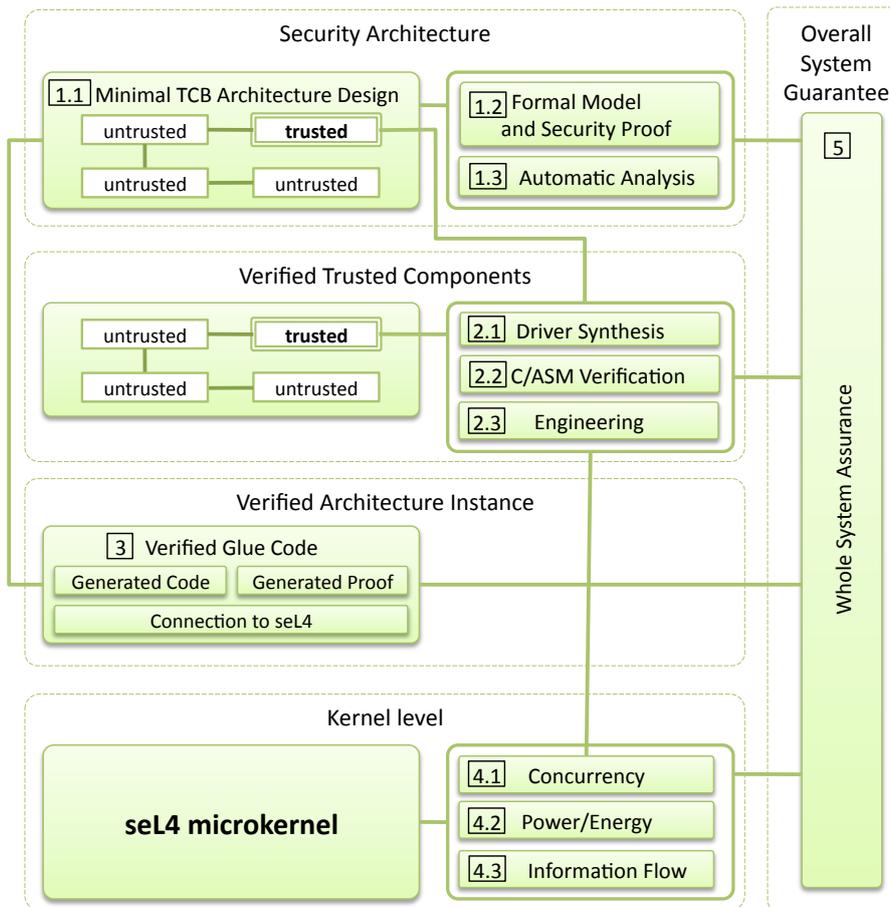
Figure 2: Research challenges and project structure.

guarantees. The formal architecture model will contain the system components, the security and fault isolation boundaries, how the components are connected, including the low-level hardware components of the platform, and how large legacy parts of the system can be isolated and later be largely ignored. The high-level design can then be used to verify that the security properties of interest hold for the architecture (assuming the components are correct for now). The high-level architectural modeling, together with model verification, is a key differentiator from similar microkernel-based work to minimize a trusted computing base [33].

The research emphasis on the architecture level is not on inventing new architecture modelling languages, but instead on using and connecting to existing ones. We will start by building on the outcome of the NICTA CAmkES project for component architectures in microkernel systems. The software engineering group at NICTA MC/ATP proposes an investigation of the connection between this language and our formal models, to existing modelling languages. This is intended to provide support for architectural quality analyses and architecturally-driven reuse. The initial main candidates are UML [10] and AADL (Architecture Analysis & Design Language) [1].

The verification of the architecture design will be complemented by automatic analysis techniques. More precisely, the MC group at CRL is developing promising theorem-proving techniques that may be applied in the context of the ERTOS-2 project. In particular, we expect the Darwin system [4], co-developed by Peter Baumgartner, to be suitable for the automated analysis of label-based security properties on the architecture level. We propose to start with selected case studies from the multi-level secure terminal design in the first year, and, if successful, fully integrate the Darwin system into the ERTOS-2 platform in the following years.

The overall output of the project at the architecture level is a class of architectures that minimise the trusted computing base, the ability to express and analyse these using existing approaches from the software architecture discipline, the ability to certify beyond doubt that the TCB has been correctly identified with respect to a particular property, and that the trustworthiness of third-party and legacy code is indeed irrelevant to the security and safety of the overall system and its critical parts.

The work load distribution is summarised in Figure 3. It is divided into the following high-level work packages.

- WP1.1: Minimal TCB architecture design, led by Kevin Elphinstone and Ihor Kuz.

- WP1.2: Formal model and security proof, led by June Andronick.

- WP1.3: Automatic analysis techniques, led by Peter Baumgartner.

Verified Trusted Components.   Having worked out how to focus effort on a small number of clearly-identified components, we now have to prove that this small trusted computing base itself establishes the desired properties. We propose investigating two complementary ways for providing verified trusted components: driver synthesis and C/ASM verification. In addition, we propose the investigation of a number of smaller engineering challenges that make the approach practical.

Driver synthesis is the automated generation of OS device drivers from formal specifications. Even with an optimal architecture that uses isolation to remove most device drivers from the trusted computing base, we anticipate that a few drivers will have to remain in the TCB, be it to provide assurance on resource availability or to enable resource sharing for performance. Driver synthesis addresses this issue. We propose the automatic synthesis of device drivers from two main sources: VHDL-level descriptions of the device internals and programming interface, and a specification of the internal operating system interfaces that a driver must conform to.

Given such specifications (which are necessary, at least in an informal form, for *any* driver development), a driver synthesis tool can generate a complete driver implementation in C. Most

Work package	Surname	First Name	Position	% of Time in WP	Total % of Time in Project
Project Leader	**Heiser**	**Gernot**	**Sen. Princ.**	**30%**	50%
WP1.1 Minimal TCB Architecture	**Kuz**	**Ihor**	**Sen. Res.**	**70%**	70%
	Elphinstone	**Kevin**	**Sen. Lec.**	**15%**	35%
	PhD3 TBD		**PhD student**	**100%**	100%
	Applicant Res. Eng. 4		**Res. Eng.**	**100%**	100%
	Zhu	**Liming**	**Res.**	**30%**	30%
	Liu	**Jenny**	**Sen. Res.**	**30%**	30%
WP1.2 Formal Model	**Andronick**	**June**	**Res.**	**70%**	70%
	Sewell	**Thomas**	**Res. Eng.**	**100%**	100%
WP1.3 Automatic Analysis	**Baumgartner**	**Peter**	**Princ. Res.**	**20%**	20%
	Applicant Res. Eng. 1		**Res. Eng.**	**50%**	100%
	PhD1 TBD		**PhD student**	**100%**	100%
WP2.1 Driver Synthesis	**Ryzhyk**	**Leonid**	**Res. Eng.**	**100%**	100%
	Cassez	**Franck**	**Visitor**	**40%**	40%
	PhD2 TBD		**PhD student**	**100%**	100%
WP2.2 C/ASM Verification	**Norrish**	**Michael**	**Sen. Res.**	**40%**	40%
	Engelhardt	**Kai**	**Sen. Lec.**	**20%**	100%
	Bourke	**Timothy**	**Res. Eng.**	**100%**	100%
	Walker	**Adam**	**Res. Eng.**	**100%**	100%
	Greenaway	**David**	**PhD student**	**100%**	100%
WP2.3 Engineering	**Chubb**	**Peter**	**Sen. Res. Eng.**	**50%**	50%
	Staples	**Mark**	**Sen. Res.**	**30%**	30%
	Applicant Res. Eng. 3		**Res. Eng.**	**100%**	100%
	Applicant Res. 2		**Res.**	**100%**	100%
WP3 Verified Glue Code	**Applicant Res. 1**		**Res.**	**70%**	70%
	Boyton	**Andrew**	**PhD student**	**100%**	100%
	Klein	**Gerwin**	**Sen. Res.**	**10%**	70%
WP4.1 Concurrency	**Engelhardt**	**Kai**	**Sen. Lec.**	**80%**	100%
	Elphinstone	**Kevin**	**Sen. Lec.**	**10%**	35%
	von Tessin	**Michael**	**PhD student**	**100%**	100%
WP4.2 Power/Energy	**Heiser**	**Gernot**	**Sen. Princ.**	**20%**	50%
	Applicant Res. Eng. 2		**Res. Eng.**	**50%**	100%
	Le Sueur	**Etienne**	**Master student**	**100%**	100%
	Carroll	**Aaron**	**PhD student**	**100%**	100%
WP4.3 Information Flow	**Elphinstone**	**Kevin**	**Sen. Lec.**	**10%**	35%
	Cock	**David**	**PhD student**	**100%**	100%
	van der Meyden	**Ron**	**Visitor**	**10%**	10%
	Applicant Res. Eng. 2		**Res. Eng.**	**100%**	100%
WP5 Overall Proof	**Klein**	**Gerwin**	**Sen. Res.**	**60%**	70%
	Applicant Res. Eng. 1		**Res. Eng.**	**50%**	100%

Figure 3: Manpower distribution over work packages

previous work on driver synthesis was in the context of hardware/software co-design for embedded microcontrollers [24], and is not directly applicable to more complex devices and operating systems that we are targeting. Previous attempts to synthesise drivers for such devices have failed to produce conclusive results [7,37]. The first-year milestone for this part is a higher-level device specification language and compiler that will translate such specifications into an abstract symbolic state machine, which can be used as an input to the driver synthesis process. The final fourth year outcome will be the full driver synthesis tool.

Manual formal verification of C and assembly code is our second main approach to provide assurance for trusted components. Here, we fully build on the work of the L4.verified project and apply techniques used for the formal verification of seL4 on the component level instead of on the kernel level. The main research challenges are extending the existing verification framework to handle concurrency and assembly instructions. The former is an open-ended research question and we propose, in the first year, to investigate only a simple, restricted subset of concurrency principles such as synchronisation primitives and message passing (seL4 IPC) in our C framework. The focus will not be on inventing completely new concurrency formalisms, but instead on integrating existing formalisms [16, 25, 26] into our current verification framework without losing its practical applicability. Assembly verification is a problem that we have already started to address in the L4.verified project. The challenge here is to extend the existing ARM assembly models into a practical, well-engineered framework that allows efficient reasoning similar to our C verification work. Both of the verification challenges above will require us to model seL4 microkernel system calls from the user perspective of a distributed, concurrent system instead of the internal kernel perspective.

The project aims to not only give assurance that existing components work, but also to provide at least a minimal engineering environment to develop them and put them together into a runnable system. This supports and enables research in the project, and is an important and non-trivial part of the project with significant challenges in itself. For the first year, we propose to instigate processes and tools for configuration management, an integrated software build environment, and to implement a complete port of Linux [19] to the seL4 kernel on the x86 platform as well as controlled access to devices (*device pass-through*) using Intel's VT-d architecture [14]. For the latter two, we have built prototypes in a related, concurrent project (due to finish in Nov'09); the work proposed here is to integrate them into the verification-enabled framework and make them into easily re-usable components that we can build on for the multi-level secure terminal.

Large-scale reuse, for example as performed in Software Product Line development, requires sophisticated configuration management. The project will explore as a research issue how component reuse and variation can be supported by the configuration management environment and by the system architecture models from work package 1.1. The configuration management and build system are also basic prerequisites for reproducibly building larger software systems on top of the microkernel. We propose a dedicated effort to build a platform that lasts for this and future projects and is of high enough quality such that it can be used easily by external collaborators.

In the longer term, the engineering part of the project will contain the implementation of the multi-level secure terminal design developed in the architecture part of the project. Connected to this implementation is the research challenge of creating tools that run under seL4/Linux to be able to set up new virtual machines with access to particular resources, to quickly start them, stop them, suspend and resume them. These are important to share resources between Linux instances over time and thus remove them from the trusted computing base.

The structure of this work package is summarised below, the work load distribution in Figure 3.

- WP2.1: Driver synthesis, led by Leonid Ryzhyk and Franck Cassez.

- WP2.2: C/ASM verification, led by Michael Norrish and Kai Engelhardt.

- WP2.3: Engineering, led by Peter Chubb.

Verified Architecture Instance.   The previous two main research challenges were finding the right architecture and assuring the small number of trusted components. Given these two, we need to use the microkernel mechanisms such that the architecture boundaries are enforced by the kernel, that the right components execute in the right compartments, and that the right communication channels are set up (and no unwanted channels can be set up).

We propose to build an automatic tool (the booter) that takes an architecture description from work package 1.1 and generates code setting up the correct microkernel policies and compartments with a series of seL4 system calls, distributing authority according to the architecture boundaries. We are currently building a simple prototype of this booter for a related, concurrent project. In the proposed project, we will extend this tool to cover the architectures investigated in work package 1.1 and more importantly, we propose to extend the tool to not just generate the necessary code, but also code-level proofs showing that the code sets up compartments according to the abstract formal semantics of the architecture model. This will make the security analysis on the abstract model meaningful and connected to the system as it is deployed. We propose not to verify the code generating tool, but instead to (automatically) verify the generated code. Our experience in the L4.verified project with such generated proofs for low-level C constructs shows that the resulting proof can be expected to be simpler as well as easier to integrate with the proofs produced by the other parts of the project.

The main work load is distributed over an existing PhD student (Andrew Boyton) whose topic is in this area for the formal side and a prospective Lab-staff researcher for the OS side of this problem. The work load is summarised again in Figure 3. This package has no sub-packages:

- WP3: Verified Architecture Instance, led by Gerwin Klein.

The kernel level.   On the microkernel level we identify three main research challenges that can be grouped into multicore issues and information flow analysis. All of these issues affect more than just the kernel level, but the research challenges are at least initially concentrated on the OS kernel.

Chip multiprocessors ("multicores") are already commonplace in the server/desktop space as well as in high-end embedded systems such as mobile phones. Our work will not be usable in practice if it cannot be applied to multicores. This requires much more than a simple extension of uniprocessor approaches. Specifically, multicore issues fall into two categories: the challenge of verifying concurrent systems, and the need to effectively manage resources, especially energy, in multicores.

The concurrency challenge requires re-visiting the verification of our fundamental building block, the seL4 microkernel. In its current version, seL4 is a uniprocessor microkernel and does not allow preemption within the kernel except from a few well-defined preemption points in long-running system calls. This allows us to simplify dealing with concurrency in the current formal proof of functional correctness of seL4. The guarantees provided by formal verification of seL4 are lost when deploying it on a multiprocessor system, unless all but one core are guaranteed to remain turned off. In this project, we will explore multicore implementation designs and verification approaches of seL4 in order to extend the desired isolation guarantees to multicore systems.

The research challenge here is to work out how best to extend seL4 to an SMP architecture such as is provided by modern multi-core embedded processors, while continuing the strong formal guarantees that the uniprocessor seL4 has. Currently we plan to tackle concurrency in multiple steps: (1) Design/implement a multicore version of seL4 which runs one instance of seL4 on each processor completely isolated from each other, i.e. without sharing of resources or having communication channels between them. After having proved that those instances

13

are bootstrapped correctly, the current uniprocessor proofs still hold for each of the instances. (2) Introduce communication/synchronisation primitives between the instances and investigate verifiability of those primitives. The solution currently favoured by OKL for their OKL4 product line falls in this category. It features a synchronisation point at kernel entry which experience suggests is appropriate for multicores in the embedded space with shared L2 caches and low inter-core latencies. We will evaluate this solution and if not tenable for our purposes can add two further steps: (3) Add the possibility of transferring resources (e.g. physical memory) between the instances and explore the consequences of that with regards to the verification done so far. (4) In addition to transferring resources, also allow free sharing of kernel data structures and investigate its effect on verifiability. Recent related work has investigated architectures for SMP kernels [30], however there has been little to no focus on verification of software on multiprocessors.

The second challenge in this area, multicore resource management, is somewhat opportunistic, in that it is not strictly required for the overall project vision, but has the potential to add to its impact by leveraging our relationships with processor manufacturers (Intel, Qualcomm, ARM), market insights/access (through OKL) as well as recent lab-based research on energy management that has led to high-profile outcomes despite being lowly resourced (one PhD student and an average of 0.2 FTE staff). We also have a good pool of research students wanting to work in this area.

Multicores add a further dimension to energy management, by providing the ability to shut down complete cores. This has the advantage of energy use scaling linearly with computing power, while for all other mechanisms (such as voltage and frequency scaling and using sleep states) this relationship is strongly sub-linear. Hence it should be the energy-management mechanism of choice in mobile systems. Multicore energy-management research is in its infancy, with very little truly applicable work available [15, 20, 32]. Multicore energy management requires dynamically shifting activities, and therefore resources, between cores. While this is enabled by the use of seL4 as a separation kernel and hypervisor, it poses challenges to verification, because it dynamically changes the execution environment of the system. There are also significant challenges in energy management that are unrelated to assurance. One stems from the likely use of manycores in the embedded space: Activities are likely to be allocated to (small sets of) dedicated cores. Such activities may run on top of a guest OS (inside a virtual machine provided by the microkernel) or on "bare hardware" (in reality also virtualised by the microkernel). These activities (or their OSes) will typically perform their own energy management. However, energy cannot be efficiently managed locally; in a mobile device, a system-wide energy-management policy must be enforced for an optimal outcome. A further challenge is that it is at present unclear whether it is possible to maintain temporal isolation concurrently with energy management.

Specific issues that will be addressed in this project include the definition of policies that are suitable at the microkernel level and that leverage (rather than interfere with) guest policies, as well as the development of models that allow a policy module to accurately estimate the effect of power-management actions on power consumption and performance. This includes managing the trade-offs involved in migrating activities between cores (to allow powering down processors) and memory banks (to allow putting memory into a sleep state), investigation of additional hardware mechanisms that would support resource management (manufacturers have expressed interest), and investigation of guest-neutral and hypervisor-neutral OS-hypervisor interfaces that can be used to implement customer-specific hypervisor-vendor and OS-vendor independent resource management. We will also investigate the interaction of energy management with real-time performance and temporal isolation, including investigation of hypervisor-level scheduling policies that support power management while maintaining real-time guarantees.

This energy part of the work package will be mostly staffed with research students, and will therefore be broken into relatively independent sup-topics to minimise the risk on the students.

The information-flow research challenge concerns two sides: analysing flows of information via

storage and timing channels, in particular providing spatial and temporal isolation [27] between components. Our previous verification efforts have focussed on the functional correctness of the seL4 kernel. We now propose to exploit the formal models we have built of the code base to analyse flows of information as well to formally prove that the kernel can be used to enforce confidentiality properties. More specifically, we propose to investigate if and how Ron van der Meyden's theory for intransitive non-interference [22] can be applied to our kernel models in an initial 6-month case study supervised by Ron van der Meyden. If successful, this case study could be extended to the full kernel over the following two years. For the temporal isolation side, we have a APA PhD scholarship applicant interested in exploring the relationship between real time guarantees needed for embedded systems, and timing-based covert channels that need to be minimized for security. There is a trade-off between accurate timing required for real time, and the fuzzy time used to minimize covert channels. The student would begin by analyzing existing promising real time scheduling algorithms in the light of covert channel bandwidth minimization. The goal is to develop practical approaches, with the relevant balance between real time accuracy and covert channel minimization.

Summarising, below the work package structure. For work load distribution we refer back to Figure 3.

- WP4.1: Concurrency, led by Kai Engelhardt.

- WP4.2: Power/energy, led by Gernot Heiser.

- WP4.3: Information flow analysis, led by Kevin Elphinstone.

Overall System Guarantee. The research challenges described so far have each been for a particular aspect of the system, covering the space necessary for providing whole-system guarantees on the code level. However, it is not enough to solve all pieces of the puzzle separately. They need to be integrated and fused together into one overall framework to combine into a single theorem stating that *this* large, complex system satisfies *this* security property on the code level.

The challenge is to build a unifying framework than can incorporate the modelling and analysis artefacts from each of the levels of the project: the existing code-based L4.verified proof, the code-level architecture instantiation, the architecture analysis conducted on the high-level transported down to a guarantee on the code level, and finally the code-level assurance for trusted components. The vision is that the whole-system framework provides a state-machine based global view of system execution similar to the L4.verified refinement setup, but potentially for concurrently operating components on multi-core machines. The system architecture can then be mapped into this framework together with the code-level semantics for components and the kernel. This makes it possible to map the overall system security property on the architecture level down to one main theorem on the state-machine model. Each of the levels in the project can then be understood as satisfying a proof obligation that is generated by this main theorem.

Instead of fully developing the framework before all other parts of the project start, we will begin the activities concurrently and run the framework as an initially independent small case study that demonstrates the feasibility of the approach after year 1 and that provides the integrating long-term formal goal statement for the other formal work packages. Without this work package, we would run the risk that the other parts of the project drift apart and end up with formally incompatible statements that do not combine into a whole-system guarantee. The work package leader, Gerwin Klein, is experienced in the integration of formal frameworks with different language and abstraction levels from the L4.verified project. He will also be involved with a small time percentage in the other formal work packages to provide the unifying perspective and push that is necessary to obtain a whole-system view.

- WP3: Overall System Guarantee, led by Gerwin Klein.

2.3   Contribution from external organisations

For external research-oriented linkages we expect further develop our existing collaboration with Galois Inc in the area of high-assurance systems. Galois Inc is a small, but widely known company in the space, and has been working closely with the NSA on formal verification and functional programming languages for high-assurance systems in the past. We have had successful past engagements with Galois Inc in developing mutually beneficial technology. They are actively publicising our technology to US defence agencies, and we expect research opportunities to arise in this space. In particular, Galois is developing verification techniques for functional programs that could provide an additional, complementary approach for trusted components and they are currently working on a static analysis tool for information flow properties on C programs that could possibly be applied to the seL4 microkernel.

We also will evaluate local collaboration opportunities with local high-assurance and security researchers and SMEs, including: Prof. Vijay Varadharajan at Macquarie University, an internationally recognized security expert; Prof. Jill Slay, University of South Australia, an internationally recognized SCADA expert; Defence Science and Technology Organisation in the area of multi-level security, and Fluffy Spider Technologies for graphics display technologies.

2.4   Risk

The main risks for the ERTOS-2 project are loss of key people, inability to meet research challenges, being overtaken by competitors, lack of market interest, and lack of interest from certification authorities.

With regards to loss of key people, we have built up a significant pool of expertise in both verification and operating systems such that no single person is a single point of failure for the whole project. Only a loss of several key people across the main research areas in this project would have a critical impact. Namely, if two or more of Gernot Heiser, Kevin Elphinstone, Gerwin Klein and Ihor Kuz would leave the project, the project would be in serious trouble. We are moderating this risk by hiring additional researchers in these areas such as we have done with June Andronick in the verification area. These positions are integrated in the resource plan and we already have prospective candidates. It will take time for these new researchers to be able to carry a similar load, but at least the impact of the risk above would be more distributed and reduced. For PhD students and research engineers, our experience in training people across domains, such as training OS people to do formal verification and vice versa, widens the group expertise and minimises the risk of drastic competency loss in any one area.

The risks of not meeting individual research challenges are inherent to research. We mitigate these risks by building on past success, ensuring that we have world class experts responsible for the key work packages, and organising the project such that the riskiest research is not core to the overall success (although it does have the capability to significantly improve on the outcomes of the project). We have identified three main risks across the research challenges. First, the full formal verification of multicore systems could turn out to be infeasible. This risk is medium to high but the impact is low, because, given the current state of the art in the area, any improvement towards partial verification would still represent significant progress that would put us in a favourable position for commercial deployment. Second, we may find out that components in the architectures of complex security-critical systems are so tightly intertwined that it is impossible to properly isolate a small trusted computing base and reduce verification effort. The impact would be important but the risk is low given the promising results of the initial work already achieved in the group. Finally, the

last important risk is that the overall system assurance model and theorem may contain too much or to little detail to be usable. The project output would then only be a formal guarantees at each level, and an informal argument about connecting them into an overall system assurance. This would of course weaken the statement and reduce the impact, but again, it would already represent a significant advance in the state of the art.

Other identified research risks are that the automatic tool Darwin does not apply in our framework, that the assembly verification fails to be practically feasible and that the proposed mechanisation of existing information flow theories does not scale. All these risks are medium to high but the impact is low, because they are not on the critical path of the project and we have alternative solutions to explore should they be needed.

With regards to competitors, the risk of our technology being overtaken is low thanks to our previous success in formally verifying the seL4 microkernel. This places us in a unique position with a head-start of several years with the only existing verified kernel upon which we can build, and unique expertise in scalable C verification.

A related risk is that our target markets do no show interest in our technology. We see this as a low risk as evidenced by the interest already shown by local and international defence related companies as well as continued interest in our research shown by other parties through OK Labs. It will be crucial, however, to both invest effort in creating further long-term market demand for our technology—by demonstrating its possibilities and convincing relevant policy makers and certification authorities of its importance—as well as investigating other potential key markets and determining their levels of interest in our technology. If we are unable to create demand in the high-assurance markets, we will still have the option to investigate opportunities for commercialising in other markets (for example, those in which certification is not required, but in which the extra dependability provided by our approach is a selling factor).

3 Commercialisation and end user impact

3.1 Opportunity & Novelty

The technology developed in this project is a general method to build complex, high-assurance embedded systems, a set of implementation and analysis tools, and a library of trusted components that can be applied in a number of verticals. The ultimate customer problem we solve is the need for secure, reliable and safe systems for an affordable price and complex feature sets. The current lack of solutions combining these conflicting interests results in markets having to make a choice between high assurance and functionality that is usually resolved towards more features, sacrificing security.

As mentioned in section 2, in the high-assurance defence computing market, for instance, the solution for dealing with sensitive data of national security is to handle such data on completely separate networks and systems, leading to multiple terminals on each desk (one for each clearance level and source network) and very awkward and inefficient work flows. Our method would allow us to build truly trustworthy, so-called multi-level secure terminals, that would make it possible to concurrently work on a highly classified document from different security categories, while referring to unclassified material (and even perform normal Internet searches) at the same time on the same terminal, with much higher assurances of security than what it is currently achievable even in less-flexible setups. Current commercial multi-level terminals build on operating systems like Windows NT, SELinux, and Trusted Solaris which are certified only to Common Criteria EAL4 (see e.g. [34]), because their trusted code base is much too large to achieve higher levels of assurance. EAL4 provides no security guarantees against knowledgeable or well-funded attackers, let alone hostile government agencies. Even Green Hills Software's recent EAL6+ microkernel certification

goes as far as what we have already achieved, as it does not prove functional correctness. Also, to our best knowledge, their formal verification does not cover the source code, but instead targets a model of the implementation. Our solution would provide assurance beyond the highest Common Criteria development requirements (EAL7), by providing formal guarantees about the source code itself. This would apply not only for the OS kernel (which we already have achieved), but for the security of the whole system.

Our solution also applies to verticals where security and reliability are required, but less critical, especially where the market pressure for features and cost reduction outweighs the high cost of current assurances processes. Examples would be the mobile phone and financial sectors, or similar high-volume markets with an emerging need for higher security, or where the cost of a software error embedded on millions of devices would be prohibitive, but not as much as currently existing high-reliability solutions. Our solution would provide dramatically enhanced security and reliability for almost no increased cost. By minimising formal methods involvement and providing a set of tools to rapidly implement highly secure and reliable systems, the ERTOS-2 project reduces the additional assurance cost of systems with millions of lines of code to well under the 60%–80% that are currently spent on testing and validation, and does so for a level of assurance that is infeasible to achieve by traditional methods.

The clearest initial market opportunity is to fill the lack of viable solutions for security in high-assurance markets like the defence area. We have strong interest from our current industry partners such as Galois in the area. According to Ed Hammersla, chief operating officer of Trusted Computer Solutions (TCS), there are in the US *between 100,000 to 200,000 people in the intelligence community who would want to use this [multi-level secure] technology. The best estimates now are 15,000 to 20,000 installed, and that's probably a high number. Any way you look at it, we are maybe at 10 percent adoption in the intelligence community and probably less than 5 percent in the broader [DOD] warfighter community [11].* There are also clear trends in E-government around the world to increase internal networking and data exchange. This is precisely the climate our solution caters to. Our technology has the potential to transform the current market into one where whole-system guarantees become a reality even when multiple components from different vendors are used. Commercial-off-the-shelf (COTS) software has been a major trend in the area for the last decade and our whole-system solution would provide the associated cost-saving benefits for governments and their contractors with increased instead of decreased security. The MILS research community has driven a similar agenda since the early 1990's, but while the concept as such is accepted and a number of advances on the theory side have been achieved, it has so far failed to materialise in the market-place with the assurance levels that are theoretically possible. This is because so far there was no viable formally verified foundation (OS kernel) and no viable, practical technology to lift the guarantees of such a foundation to guarantees about the whole system. We have achieved the first, this proposal is about providing the second.

The outcomes described above could be commercialised by forming a startup company that applies the technology to build a product, the technology could be licensed to existing partners (e.g. OKL), or the technology could be commercialised directly in the forms of tools. We also propose for the first year to analyse the defence market closer and to investigate which opportunities exist in alternative verticals. These include high- to medium assurance markets such as automotive, bio-medical, avionics, SCADA and even the cloud computing sector. The latter is relevant, because we expect our solutions for enforcing security policies in multicore systems to generalise to wider distributed systems. For instance, a verified AppStore that can enforce access control policies over millions of iPhone devices could become a possibility in the medium term. OKL's developing engagements with customers in security-sensitive markets, such as point-of-sales terminals, defence, automotive and avionics, provide a possible conduit for the commercialisation of the technology developed in this project. OKL is frequently approached by representatives of verticals the company is currently not servicing, but that could represent a potential market for the technologies in this

project. Working together with OKL in this area will provide us with already interested and targeted contacts.

Our initial IP strategy for this project is to investigate the potential for core patents in each of the major research challenges described in section 2 and to hold off publication on more than general project descriptions until these are clarified. For the remaining areas we will follow a similar approach to the previous ERTOS projects with a balance between open source/publications for maximum impact and key confidential parts for keeping competitive barriers. Our first commercialisation milestone after month 6 is to complete a formal IP strategy for explicitly managing the publications/confidentiality trade-off.

An initial search of the patent space relating to multilevel secure systems and networks, as well as technologies relating to secure kernels and separation kernels, shows that there are no significant blocking patents. While patents do exist in these fields, they relate to technologies that use specific hardware techniques to achieve isolation between security levels or mutually untrusted systems, or different software approaches to achieving separation in operating system kernels. We are not aware of any patents covering the formal verification of operating systems or whole-system guarantees.

3.2 Capability

The critical success factors of the project are our capability to reduce the trusted computing base of such systems on the architecture level, to formally connect the security architecture to the verified code base and to give assurance about individual trusted components. While there are significant research challenges in all three factors, we can bring strong experience and a proven track record to bear on these challenges.

The main risks that could destroy the opportunity are failing on the critical research challenges, competitors being successful in watering down government requirements and expectations on security assurance, and a competitor with significant resources like Microsoft or VMWare duplicating an extremely high-assurance version of a system similar to our verified microkernel. Both companies have so far concentrated their business on different markets. While we estimate at least a 2–3 years head start that we widen with the presented research agenda, these competitors would theoretically have the resources to fully buy out our current team (which we might as well count as a commercialisation success). Another risk are barriers to enter the certification market internationally on a broader scale with our new technology, even if that technology is just the full realisation of the initial vision of multi-level secure systems. We believe that our current engagement with local defence in Australia and Galois in the US will be conducive to reducing this risk.

Our project team has a proven track record in the research as well as in the commercialisation area and in engaging with industry. The ERTOS group is home to two of NICTAs initial startup companies, one of them (OKL) directly evolved from UNSW and NICTA research, and all our completed NICTA research projects have core IP commercially licensed to OKL. We plan to work together with OKL to bolster our network of market contacts in the first year and to complement the current team focus, which for the first years of the project is (appropriately) emphasising the research side.

3.3 Industry collaboration

Our current market-oriented linkages include Intel, the Department of Defence, Galois Inc, and Open Kernel Labs (OKL). We see OKL as a partner for future commercialisation efforts, providing us with direct access to customers and the market relevant to microkernel-based systems, and, equally important, market requirements. Our experience so far has shown that a productive

partnership useful for both sides is achievable. Our research agenda is a good fit to OKL's roadmap, as initial market feedback indicates that a verified kernel on its own will not go very far; the technology will be valued more once guarantees can be made on the whole system—exactly what the current project attempts. Therefore, the project will amplify the value of OKL's existing verification IP.

Although OKL will be a very valuable partner, we are not tied to them in terms of IP developed in this project. We acknowledge that the IP situation with OKL needs to be handled with care to not complicate it unnecessarily. However, the IP already assigned to OKL is strictly complementary to what we will develop on top of the kernel in this project. The strongest picture is continuing our close collaboration with OKL, but it is not our only option forward.

Further commercial linkages are local defence, Galois, and Intel, plus an emerging partnership with embedded processor vendor ARM. We currently receive funding from local defence. Galois Inc is an important strategic research partner we have collaborated with in the past. The company has about 40 employees and is a US-based small business working closely with the NSA and other Dept. of Defense agencies on high-assurance systems and components since 1999. Galois consequently has invaluable experience and contacts in our target market, again with a complementary and compatible roadmap. Finally, we intend to extend the current engagement with Intel who presently provide funding for our driver work.

This concludes the commercialisation section. While still at an early stage, we believe to have identified a clear present and increasing opportunity for commercialising the proposed research with a potential for truly global impact.

4 National Benefits for Australia

This proposal is aligned with National Research Priority 4 (Safeguarding Australia), specifically the priority goal "Transformational defence technologies". Current defence intelligence systems are not widely network-connected if they are to be secure for highly classified work, significantly inhibiting their use and even lagging behind normal office use. The advances described in this proposal would enable highly secure systems to become significantly more connected and increase their efficiency dramatically. This is an active area of research in allied and non-allied countries. Local defence has already expressed interest in our current work. Systems built according to the methods in this proposal have the potential to push Australia's capabilities far beyond those of the rest of the world.

Besides defence, the present proposal is also clearly aligned with the recently published government's research priority areas, specifically with the area of embedded systems, which are central to this proposal. Furthermore the formal guarantees enabled by our work will also be beneficial in areas where certification is not required, but in which security and dependability are critical, such as online commerce and the digital economy. This provides a secondary alignment with National Research Priority 3 (Frontier Technologies for Building and Transforming Australian Industries) priority goal "Smart information use".

In all these areas (defence, embedded systems, and digital economy) Australia has a chance to play a globally leading role.

5 Research Portfolio Balance and Collaboration

5.1 Fit into strategic plans

The proposed project is the direct implementation of the RSG-approved second ERTOS Strategic Plan which itself is a major corner stone of the overall Embedded Systems Theme strategic plan (research area 6.1 in the plan). The project leader as well as the work package leaders Kevin Elphinstone and Gerwin Klein are identified as NICTA key research leaders in the Embedded Systems strategic plan. The proposal is also aligned with the Managing Complexity theme strategic plan. In particular, it drives the logic and formal methods agenda of the plan in interactive verification (e.g. on the C level) as well as fully automated methods (e.g. in security analysis). The MC strategic plan explicitly mentions the work package leaders Peter Baumgartner, Michael Norrish and Gerwin Klein as key NICTA research leaders in the area.

5.2 Collaboration

Within NICTA, our research plan includes four main groups in the Embedded Systems (ES) and Managing Complexity (MC) themes: the core ERTOS group in ES, the Formal Methods and Logics groups in MC in Kensington and Canberra, and the Software Engineering group at ATP. We also plan to strengthen our existing linkages to UNSW by involving researchers with capabilities complementary to the NICTA team. Specifically, we will be working with Ron van der Meyden on security architectures for information flow and with Kai Engelhardt on concurrency at the OS level and in component systems. In later stages we intend to involve Carroll Morgan on compositional calculi for information flow properties and Manuel Chakravarty on using the functional programming language Haskell on the OS level in microkernel environments. These activities concern security properties of the overall system, the integration of individual pieces into whole-system assurance, and methods for producing high-assurance components at reasonably low cost.

We have collaborated successfully with these partners in the past and have shown that this close mix of formal verification, OS, and software engineering expertise gives us a unique competitive advantage over research groups over comparable or larger size.

6 Education

Our current PhD students Rafal Kolanski, Andrew Boyton, Michael von Tessin, and Aaron Carroll, our Master's student Etienne Le Sueur, as well as three recent prospective PhD applicants (to commence APA in July), are part of our longer-term research strategy and are fully integrated into the project from the onset. Leonid Ryzhyk, a PhD student in ERTOS who is currently finishing, will be a work package leader and carry over his successful device driver research into the ERTOS-2 project.

We will continue to recruit PhD students since the research challenges we face offer excellent opportunities for PhD research. Our strategy is to provide PhD students with topics that enhance the work being done, but are not on the most critical path. Thus, while PhD students form an integral part of the project team, we nevertheless provides security for both the project and the student.

Given our collaboration with external organisations like Intel and ARM, students will be presented with a range of opportunities to engage in industrial internships and academic exchanges (such as TU Dresden, ETH Zurich, and TU Munich). The group has a strong track record of placing students for internship in prestigious industrial research labs (IBM Watson, HP Labs, Intel Research),

and this has frequently impacted their research by exposing them to new problems. It has also significantly contributed to the international reputation of the NICTA team.

The team also has a strong track record of enhancing undergraduate education at UNSW through coursework teaching, and the project leader holds the John Lions Chair of Operating Systems at UNSW, which ensures a close fit between the project research and the operating systems focus at UNSW. Specifically the teaching of Operating Systems, Advanced Operating Systems, Distributed Systems, Advanced Topics in Software Verification, and Foundations of Concurrency has greatly increased student interest in (and preparedness for) systems research at NICTA, and has created a strong pipeline of prospective PhD students, besides satisfying OKL's demand for top-class engineers. The skills of the UNSW Systems graduates are also underscored by their employment in leading multinationals (Apple, VMware, Google). UNSW is now seen as one of the world's top sources of Systems graduates, and this position is likely to strengthen with the contributions to teaching from the new researchers we plan to hire.

References

[1] AADL: Architecture analysis & design language. http://la.sei.cmu.edu/aadl/currentsite/, 2009.

[2] F. A. Administration. Special conditions: Boeing model 787-8 airplane; systems and data networks security—isolation or protection from unauthorized passenger domain systems access. http://cryptome.info/faa010208.htm, January 2008.

[3] J. Alves-Foss, P. W. Oman, C. Taylor, and S. Harrison. The MILS architecture for high-assurance embedded systems. *International Journal on Embedded Systems*, 2:239–247, 2006.

[4] P. Baumgartner, A. Fuchs, and C. Tinelli. Darwin: A theorem prover for the model evolution calculus. In S. Schulz, G. Sutcliffe, and T. Tammet, editors, *Proceedings of the 1st Workshop on Empirically Successful First Order Reasoning (ESFOR'04), Cork, Ireland, 2004*, Electronic Notes in Theoretical Computer Science. Elsevier, 2004.

[5] C. Boettcher, R. DeLong, J. Rushby, and W. Sifre. The MILS component integration approach to secure information sharing. In *27th IEEE/AIAA Digital Avionics Systems Conference (DASC)*, St. Paul MN, October 2008.

[6] C. Boettcher, R. DeLong, J. Rushby, and W. Sifre. The MILS component integration approach to secure information sharing. In *Proceedings of the 27th IEEE/AIAA Digital Avionics Systems Conference*, St Paul, NM, USA, October 2008.

[7] V. Chipounov and G. Candea. Reverse-engineering drivers for safety and portability. In *Proceedings of the 4th Workshop on Hot Topics in System Dependability*, San Diego, CA, USA, December 2008.

[8] C. Criteria. Common criteria for information technology security evaluation. http://www.commoncriteriaportal.org/, September 2007. Version 3.1.

[9] K. Elphinstone, G. Klein, P. Derrin, T. Roscoe, and G. Heiser. Towards a practical, verified kernel. In *Proceedings of the 11th Workshop on Hot Topics in Operating Systems*, pages 117–122, San Diego, CA, USA, May 2007.

[10] O. M. Group. OMG unified modeling language (OMG UML), infrastructure: Version 2.2. http://www.omg.org/spec/UML/2.2/Infrastructure/PDF/, 2009.

[11] E. Hammersla. Secure sharing of sensitive data. Government Computer News, http://gcn.com/Articles/2008/12/15/Ed-Hammersla-Secure-sharing-of-sensitive-data.aspx?Page=2, December 2008.

[12] G. Heiser, K. Elphinstone, I. Kuz, G. Klein, and S. M. Petters. Towards trustworthy computing systems: Taking microkernels to the next level. *ACM Operating Systems Review*, 41(4):3–11, July 2007.

[13] M. A. Hillebrand and W. J. Paul. On the architecture of system verification environments. In *Hardware and Software: Verification and Testing*, volume 4899 of *LNCS*, pages 153–168, Berlin, Germany, 2008. Springer.

[14] Intel Corp. *Intel Virtulization Technology for Directed I/O*, September 2007. URL http://download.intel.com/technology/computing/vptech/Intel(r)_VT_for_Direct_IO.pdf.

[15] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi. An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. In *39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, December 2006.

[16] C. B. Jones. Tentative steps towards a development method for interfering programs. *ACM Transactions on Programming Languages and Systems*, 5(4):596–619, 1983.

[17] G. Klein. Operating system verification — an overview. *Sādhanā*, 34(1):27–69, February 2009.

[18] I. Kuz, Y. Liu, I. Gorton, and G. Heiser. CAmkES: A component model for secure microkernel-based embedded systems. *Journal of Systems and Software Special Edition on Component-Based Software Engineering of Trustworthy Embedded Systems*, 80(5):687–699, May 2007.

[19] B. Leslie, C. van Schaik, and G. Heiser. Wombat: A portable user-mode Linux for embedded systems. In *Proceedings of the 6th Linux.Conf.Au*, Canberra, April 2005.

[20] J. Lieder. Energy-efficient scheduling for multi-core processors. Diploma thesis, System Architecture Group, University of Karlsruhe, Germany, November 2008.

[21] LinuxDevices.com. Linux thin client software certified "Secret". `http://www.linuxdevices.com/news/NS6191784180.html`, July 2008.

[22] R. van der Meyden. Architectural refinement and notions of intransitive noninterference. In *International Symposium on Engineering Secure Software and Systems*, number 5429 in LNCS, Leuven, Belgium, February 2009.

[23] Y. Nakamoto. Research on dependable OS for integrated services of control and multimedia in vehicles. Seminar at NICTA, July 2008.

[24] M. O'Nils, J. Öberg, and A. Jantsch. Grammar based modelling and synthesis of device drivers and bus interfaces. In *Proceedings of the 24th Euromicro Conference*, Washington, DC, USA, 1998.

[25] S. Owicki and D. Gries. An axiomatic proof technique for parallel programs. *Acta Informatica*, 6:319–340, 1976.

[26] L. Prensa Nieto and J. Esparza. Verifying single and multi-mutator garbage collectors with Owicki/Gries in Isabelle/HOL. In M. Nielsen and B. Rovan, editors, *Mathematical Foundations of Computer Science (MFCS 2000)*, volume 1893 of *LNCS*, pages 619–628. Springer-Verlag, 2000.

[27] J. Rushby. Partitioning for safety and security: Requirements, mechanisms, and assurance. NASA Contractor Report CR-1999-209347, NASA Langley Research Center, June 1999. Also to be issued by the FAA.

[28] J. M. Rushby. Design and verification of secure systems. In *Proceedings of the 8th ACM Symposium on Operating Systems Principles*, pages 12–21, 1981.

[29] N. Schirmer, M. Hillebrand, D. Leinenbach, E. Alkassar, A. Starostin, and A. Tsyban. Balancing the load — leveraging a semantics stack for systems verification. *JAR, special issue on Operating System Verification*, 2009. To appear.

[30] A. Schüpbach, S. Peter, A. Baumann, T. Roscoe, P. Barham, T. Harris, and R. Isaacs. Embracing diversity in the Barrelfish manycore operating system. In *Proceedings of the 1st Workshop on Managed Many-Core Systems*, Boston, MA, USA, June 2008.

[31] J. S. Shapiro. Understanding the Windows EAL4 evaluation. *IEEE Computer*, 36:103–105, February 2003.

[32] J. Shirako, N. Oshiyama, Y. Wada, H. Shikano, K. Kimura, and H. Kasahara. Compiler control power saving scheme for multi core processors. *Languages and Compilers for Parallel Computing*, 4339/2006, 2006.

[33] L. Singaravelu, C. Pu, H. Härtig, and C. Helmuth. Reducing TCB complexity for security-sensitive applications: Three case studies. In *Proceedings of the 1st EuroSys Conference*, pages 161–174, Leuven, Belgium, April 2006.

[34] Trusted Computer Systems. SecureOffice Trusted Workstation. http://www.trustedcs.com/documents/SOTrustedWorkstationSolTOv.pdf, September 2005.

[35] H. Tuch, G. Klein, and G. Heiser. OS verification — now! In *Proceedings of the 10th Workshop on Hot Topics in Operating Systems*, pages 7–12, Santa Fe, NM, USA, June 2005. USENIX.

[36] The Verisoft XT project. http://www.verisoftxt.de/StartPage.html.

[37] S. Wang, S. Malik, and R. A. Bergamaschi. Modeling and integration of peripheral devices in embedded systems. In *Proceedings of the 40th ACM/IEEE Conference on Design, Automation and Test in Europe*, 2003.