KTH Security Workshop 19 October 2017

Location: KTH Library, room Lallerstedt

- Program -

9.30 - 10.15: Gustavo Betarte, Department of Computer Science, Facultad de Ingeniería, Universidad de la República, Uruguay
Title: A certified reference validation mechanism for the permission model of Android
Abstract:
Android embodies security mechanisms at both OS and application level. In this platform application security is built primarily upon a system of permissions which specify restrictions on the operations a particular process can perform. The critical role of these security mechanisms makes them a prime target for (formal) verification. We have developed an idealized model of a reference monitor of the novel mechanisms of Android 6 (and further), where it is possible to grant permissions at run time. Using the programming language of the proof-assistant Coq we have developed a functional implementation of the reference validation mechanism and certified its correctness with respect to the specified reference monitor. Several properties concerning the permission model of Android 6 and its security mechanisms have been formally formulated and proved. Applying the program extraction mechanism provided by Coq we have also derived a certified Haskell prototype of the reference validation mechanism. In this talk we shall present the formalized idealized model, discuss the proven properties and describe the implementation of the certified reference monitor.

10.15 - 10.45: Cyrille Artho, KTH/TCS
Title: Verifying Nested Lock Priority Inheritance in RTEMS with Java Pathfinder.
Abstract:
Scheduling and synchronization algorithms for uniprocessor real-time systems benefit from the rich theory of schedulability analysis, and yet translating these algorithms to practical implementations can be challenging. This paper presents a Java model of the priority inheritance protocol for mutual exclusion, as implemented in the RTEMS open-source

real-time operating system. We verified this model using Java Pathfinder to detect potential data races, deadlocks, and priority inversions. JPF detected a known bug in the RTEMS implementation, which we modified along with the Java model. Verification of the modified model showed the absence of data races, deadlocks, and established nine protocol-specific correctness properties.

10.45 - 11.15: Break

11.15 - 12.00: Boris Köpf, IMDEA, Spain
Title: Static Quantification of Timing Side Channels
Abstract:
Today's execution platforms employ a wide variety of techniques for minimizing the consumption of resources such as time, memory, and energy. While these techniques are indispensable for achieving competitive performance, they can pose a serious threat to security: By reducing the resource consumption on average (but not in the worst case), they introduce side channels that can be used for recovering private information about users, or even cryptographic keys.

In this talk I present a line of work on quantifying time-based side channels using static program analysis techniques, together with an outlook on how the results of the analysis can be leveraged in the context of cryptographic and economic theories.

12.00 - 12.30: Christoph Baumann, KTH/TCS
Title: Compositional Verification of Security Properties for Embedded Execution Platforms
Abstract:
The security of embedded systems can be dramatically improved through the use of formally verified isolation mechanisms such as separation kernels, hypervisors, or micro-kernels. For trustworthiness, particularly for system level behavior, the verifications need precise models of the underlying hardware. Such models are hard to attain, highly complex, and proofs of their security properties may not easily apply to similar but different platforms. This may render verification economically infeasible. To address these issues, we propose a compositional top-down approach to embedded system specification and verification, where the system-on-chip is modeled as a network of distributed automata communicating via paired synchronous message passing. Using abstract specifications for each component allows to delay the development of detailed models for cores, devices, etc., while still being able to verify high level security properties like integrity and confidentiality, and soundly refine the result for different instantiations of the abstract components at a later stage. As a case study, we apply this methodology to the verification of information flow security for an industry scale security-oriented hypervisor on the ARMv8-A platform. The hypervisor statically assigns (multiple) cores to each guest system and implements a rudimentary, but usable, inter guest communication discipline. We have completed a pen-and-paper security proof for the

hypervisor down to state transition level and report on a partially completed verification of guest mode security in the HOL4 theorem prover.

12.30 - 14.00: Lunch

14.00 - 14.45: Deepak Garg, Max Planck Institute
Title: Enforcing confidentiality in low-level code with active attacks
Abstract:
This work-in-progress talk considers the problem of enforcing confidentiality in low-level applications (e.g., those written in C) in the presence of an active adversary. Static approaches, such as information flow type systems, are inadequate since a C-like language does not provide basic guarantees of memory safety and control flow integrity. Dynamic approaches have a high runtime overhead. Our approach strikes a middle ground by relying on an instrumenting compiler. The programmer marks secret data by writing lightweight annotations on top-level definitions in the source code. The compiler relies mostly on static flow analysis coupled with lightweight runtime instrumentation, a custom memory layout and custom control-flow integrity checks to prevent data leaks even in the presence of active low-level attacks. Additionally, using a machine code verifier, we can remove the compiler itself from the trusted computing base. The talk will provide a high-level overview of the difficulties, an implementation as part of the LLVM toolchain and preliminary evaluation results.

14.45 - 15.15: Roberto Guanciale, KTH/TCS
Title: Protecting Instruction Set Randomization from Code Reuse Attacks
Abstract:
Instruction Set Randomization prevents code injection by randomizing the instruction encoding used by programs, thus preventing an attacker from preparing a payload that can be injected in a victim. Here, we show that code-reuse attacks can be used to circumvent existing ISR techniques and we demonstrate these attacks on an ARMv7 CPU that has been extended with ISR support. To counter this treat, we propose a new ISR that has not the same vulnerabilities of the existing solutions and has moderate decryption cost, no need of additional memory per instruction, and efficient random access to the encrypted code. These properties enable efficient hardware implementation of our solution. In order to evaluate our proposal, we implement the new ISR in a hardware simulator and we compare its overhead with respect to existing ISR approaches.

15.15 - 15.45: Philipp Haller, KTH
Title: Retrofitting Types for Isolation and Affinity
Abstract:
Isolation and sandboxing are crucial for enforcing security properties. Expressing and ensuring isolation in imperative object-oriented languages is challenging due to pervasive

aliasing. The last decade has seen important advances in type systems for aliasing control. However, their large-scale adoption has turned out to be a surprisingly difficult challenge. While new language designs show promise, they do not address the need of isolation and sandboxing in existing languages.

This talk presents a new approach to isolation and affine types in an existing, widely-used language, Scala. The approach is unique in the way it addresses some of the most important obstacles to the adoption of type system extensions for aliasing control. First, adaptation of existing code requires only a minimal set of annotations. Only a
single bit of information is required per class definition. Surprisingly, this information can be provided by the object-capability discipline. We formalize our approach as a type system and prove key soundness theorems. The type system is implemented for the full Scala language, providing a sound integration with Scala's local type inference. Finally, we empirically evaluate the conformity of existing Scala open-source code to the object-capability discipline on a corpus of over 75,000 LOC.

15.45 - 16.15: Break

16.15 - 16.45: Musard Balliu, Chalmers
Title: We are Family: Relating Information-Flow Trackers
Abstract:
While information-flow security is a well-established area, there is an unsettling gap between heavyweight information-flow control, with formal guarantees yet limited practical impact, and lightweight tainting techniques, useful for bug finding yet lacking formal assurance. This paper proposes a framework for exploring the middle ground in the range of enforcement from tainting (tracking data flows only) to fully-fledged information-flow control (tracking both data and control flows). We formally illustrate the trade-offs between the soundness and permissiveness that the framework allows to achieve. The framework is deployed in a staged fashion, statically embedding a dynamic monitor, being parametric in security policies, as they do not need to be fixed until the final deployment. This flexibility facilitates a secure app store architecture, where the static stage of verification is performed by the app store and the dynamic stage is deployed on the client. To illustrate the practicality of the framework, we implement our approach for a core of Java and evaluateit on a use case with enforcing privacy policies in the Android setting. We also show how a state-of-the-art dynamic monitor for JavaScript can be easily adapted to implement our approach.

16.45 - 17.15: Oliver Schwarz, SICS
Title: On Ice Cream, Santa Claus, and a Hypervisor -- Information Flow Properties in Instruction Set Architectures
Abstract:
For the verification of system software, information flow properties of the instruction set architecture (ISA) are essential. They show how information propagates through the

processor, including sometimes opaque control registers. Thus, they can be used to guarantee that user processes cannot infer the state of privileged system components, such as secure partitions. This talk discusses two approaches on how to obtain and verify such properties in an interactive theorem prover. The objective is to do so in a (semi-)automatic manner and with guaranteed soundness and accuracy

17.15 - 17.35: Jonas Haglund, KTH/TCS
Title: Isolated Networking in Embedded Systems
Abstract:
Embedded systems can be found in many important situations (e.g. critical infrastructure), thus making security in embedded systems critical. PROSPER has addressed this security need by developing and verifying a hypervisor that establishes isolation among several software systems. The hypervisor is capable of hosting a Linux guest alongside bare-metal security services, preventing malware in Linux from interfering with the security services. This hypervisor has no software support for a Network Interface Controller (NIC), thus preventing Linux to access the Internet. If Linux is allowed to configure the NIC arbitrarily, Linux could use the NIC to interfere with the security services via the NICs abilities to access memory.

In this talk it is described how the Internet access limitation is removed, enabling Linux to access the Internet. The work consists of two parts. The first part is an implementation of a NIC-security layer in the hypervisor that intercepts all reconfigurations of the NIC made by Linux. This NIC-security layer prevents the NIC from accessing memory not belonging to Linux. The second part concerns formal verification of the isolation properties of the extended system. This second part includes a formal model of the NIC and a high-level approach for verifying the isolation properties. The verification approach applies the simulation proof method on a system model secure by design and a system model describing the actual system implementation.

17.35 - 17.55: Andreas Lindner, KTH/TCS
Title: Sound transpilation from binary to machine-independent code
Abstract:
Over the recent years a number of binary analysis platforms have emerged that have successfully expanded the scope of many verification and static analysis techniques to the domain of binary-level code. These platforms have to handle the complexity and heterogeneity of modern instruction set architectures. For this reason, they typically introduce hardware-intermediate representations and operate on these rather than the instruction sets themselves. However, drawing sound conclusions about the binary-level code from the analysis requires to trust the correctness of the translation from binary code to the intermediate language. Achieving a high degree of trust is challenging since this transpilation must correctly handle all the side effects of the instructions, multiple instruction encoding (e.g. ARM Thumb), variable instruction length (e.g. Intel), and potentially more. We address the issue of trust in a given translation of a binary program by (i) formally modeling one such

intermediate language in the interactive theorem prover HOL4 and (ii) by implementing a proof-producing transpiler. The tool translates ARMv8 programs to the intermediate language by using a formal model of the ARMv8 ISA and generates a HOL4 proof that demonstrates the correctness of the translation in the form of a simulation theorem. The transpilation procedure itself is standard. Proof production is more challenging and not implemented in any existing binary analysis platform. We propose a method along with a HOL4 implementation for generating the required theorems first for expressions extracted from the ISA model, then for a single binary instruction, and finally for whole programs. We also briefly show how the transpiler theorems can be used to transfer properties verified on the intermediate language to the binary code. This work assumes sequential application code and therefore reusing our results for parallel or system level code requires proper handling.

18.30 - ??

Workshop dinner at Rydbergs Bar och Matsal, Drottninggatan 88, Stockholm