ABSTRACT
A trusted execution environment (TEE) is a secure processing environment that is isolated from the “normal” processing environment where the device operating system and applications run. The first mobile phones with hardware-based TEEs appeared almost a decade ago, and today almost every smartphone and tablet contains a TEE like ARM TrustZone. Despite such a large-scale deployment, the use of TEE functionality has been limited for developers. With emerging standardization this situation is about to change. In this tutorial, we explain the security features provided by mobile TEEs and describe On-board Credentials (ObC) system that enables third-party TEE development. We discuss ongoing TEE standardization activities, including the recent Global Platform standards and the Trusted Platform Module (TPM) 2.0 specification, and identify open problems for the near future of mobile hardware security.

Categories and Subject Descriptors
K.6.5 [Management of Computing and Information Systems]: Security and Protection

Keywords
Trusted execution environments, mobile devices

1. INTRODUCTION
A trusted execution environment (TEE) is a secure, integrity-protected processing environment, consisting of processing, memory and storage capabilities. It is isolated from the “normal” processing environment, sometimes called the rich execution environment (REE), where the device operating system and applications run. TEEs can make it possible to build REE applications and services with better security and usability by partitioning them so that sensitive operations are restricted to the TEE and sensitive data never leave the TEE. In our daily lives, we encounter more and more services that use dedicated hardware tokens to improve their security: one-time code tokens for two-factor authentication, wireless tokens for opening doors in buildings or cars, tickets for public transport, and so on. Mobile devices equipped with TEEs have the potential for replacing these many tokens thereby improving the usability for users while also reducing the cost for the service providers without hampering security.

Chances are that the mobile device in your pocket sports a hardware-based TEE. Chances are, too, that you have not come across too many applications that actually make use of TEE functionality. In this tutorial, we explain why this situation came to pass and what the future may hold.

Security in mobile world had a very different trajectory compared to the world of personal computers. Various stakeholders had strict security requirements, some of which date back to two decades ago, right at the beginning of the explosion of personal mobile communications. Standardization requirements like ensuring that the device identifier will resist manipulation and change, regulatory requirements like ensuring secure storage for radio frequency parameters, business requirements like ensuring that subsidy locks, and end user expectations (e.g., no blue screen of death) incentivized mobile device manufacturers, chip vendors and platform providers to design and deploy hardware and platform security mechanisms for mobile platforms from early on.

Hardware-based TEEs were seen as essential building blocks in meeting these requirements. The first mobile phones with hardware-based TEEs appeared almost a decade ago, and today almost every smartphone and tablet contains a TEE like ARM TrustZone, along with software platform security mechanisms.

Despite such a large-scale deployment, the use of TEE functionality has been largely restricted to its original intended uses. There has been no widely available means for application developers to benefit from existing TEE functionality. Fortunately, with emerging standardization this situation is about to change.

In this tutorial, we explain the security features provided by TEEs and describe On-board Credentials (ObC), a system that we developed at Nokia Researcher Center for safely opening up access to TEE functionality for application developers. We demonstrate the possibilities of TEEs with a programming example using the ObC system, discuss ongoing TEE standardization activities, including the recent Global Platform standards and the Trusted Platform Module (TPM) 2.0 specification, and identify open research questions for the near future of mobile hardware security.
2. TUTORIAL DESCRIPTION

2.1 Structure and Length

The total length of the tutorial is three hours. The tutorial consists of the following parts:

1. A look back. We start the tutorial with a brief historical background on development of mobile security. We explain the business, regulatory, and end user requirements that have steered introduction of platform security features to mobile devices over the past two decades. (15 minutes)

2. Mobile hardware security. We proceed by explaining typical mobile hardware security functionality at a conceptual level. We describe mechanisms present in a typical mobile TEE, including secure boot, secure storage, isolated execution, immutable device identifiers and devices authentication. We introduce ARM TrustZone architecture [1] that is widely deployed in a majority of existing smartphones, and explain how TrustZone can be used to implement many of the previously described hardware security mechanisms in practice. (30 minutes)

3. Application development and ObC. The current standardized and de-facto hardware security APIs (e.g., PKCS-11 [4] and JSR-177 [5]) allow secure storage of cryptographic keys and execution of pre-defined cryptographic computation. To take full advantage of the isolated execution capabilities of mobile TEEs different kinds of API abstraction are needed. We describe On-board Credentials (ObC) system [3] that we have developed at Nokia Research Center. The ObC system serves as an example of an architecture that allows third-parties to develop and deploy credentials on TrustZone enabled devices. We explain the ObC architecture, its open provisioning model, developer interfaces, and a few example applications. Both the ObC system and applications built on top of it have been deployed on recent Nokia Lumia device models. We provide a brief demonstration of ObC application development in practice. (10 minutes)

Break. (15 minutes)

4. Emerging standardization. We explain recent and on-going standardization activities regarding mobile TEEs. We briefly explain the set of APIs that Global Platform [2] has specified for usage of hardware-security functionality in mobile devices. After that, we focus in more detail on the recently published Trusted Platform Module (TPM) 2.0 standard [6]. We especially explore the new authorization model introduced in TPM 2.0 and examine its potential in use cases like secure boot and provisioning. (60 minutes)

5. Summary and a look ahead. We end the tutorial with a brief summary and identify open issues and research questions for mobile hardware security in near future. (10 minutes)

2.2 Level of Detail

The first two parts of the tutorial will provide the audience a background information on mobile hardware security. We describe the historical development, the common hardware-security features, and the TrustZone architecture in a fairly high level. After that we provide more detailed treatment on two topics: application development and standardization. In these parts the tutorial gets more technical: we explain the ObC system architecture and provisioning model in detail, and, for example, cover individual TPM 2 instructions and data structures to explain the functionality of this emerging security standard. The ObC demo part illustrates practical TEE development on a programming level.

2.3 Intended Audience

This tutorial is intended for researchers and practitioners interested in mobile security in general and development of novel hardware-security services in particular. Successful following of the first half (parts 1-3) of the tutorial does not require any prior knowledge of mobile hardware security architectures. Basic background knowledge on mobile operating system security (e.g., Android OS security model) or embedded system security (e.g., smart card security) will help the listener to place some of the the explained concepts to context, but such background knowledge is not mandatory for following the tutorial. Basic background knowledge of mobile application development is useful for following the ObC demo part, but not mandatory.

In the second half of the tutorial (part 4) we discuss emerging hardware security standards, especially the recently introduced TPM 2.0 specification. To follow the standardization part, some background knowledge of basic trusted computing basic concepts, such as TPM commands and data structures, is useful. Our intention is to focus on the concepts and features that are new in TPM 2.0 specification, and we do not to give a complete description of TPM 1.2 as a background.

After the tutorial, the audience should have a good overall understanding on the security mechanisms that TEEs provide in current mobile devices, and what type of applications one can develop for TEEs. The audience should also have a basic understanding on recent and currently on-going standardization regarding mobile TEEs. These standardized interfaces are likely to make TEE programming possible for third-party developers, and the purpose of our tutorial is to raise the awareness of such emerging TEE development possibilities both in the research community as well as among practitioners.

3. REFERENCES


1. ABSTRACT

Model checking [5] is an automated algorithmic technique for exhaustive verification of systems, described as finite state machines, against temporal logic [9] specifications. It has been used successfully to verify hardware at an industrial scale [6]. One of the most successful variants of model checking is Bounded Model Checking (BMC) [2] which leverages the power of state-of-the-art satisfiability (SAT) \(^1\) and satisfiability modulo theory (SMT) \(^2\) to push the boundaries of automated verification. Like model checking, BMC was developed originally for hardware, but has since been extended and applied successfully to verify sequential [4], multi-threaded [1, 10], as well as real-time software [3].

A key benefit of BMC-based software model checkers, such as cbmc [4], is that they are able to handle bit-level semantics of programs precisely. Thus, they are able to detect errors due to integer overflows, and prove correctness of programs that use bit-level operations, without reporting false warnings, or missing bugs. This makes BMC ideal for verifying high-integrity software, where the cost of failure is substantial. Indeed, cbmc has been used to verify a wide variety of low-level safety and security-critical systems, such as co-pilots [8], OS schedulers [7], and hypervisors [11] (see http://www.cprover.org/cbmc/applications.shtml for a more expansive list).

This tutorial will provide an introduction to BMC, its underlying technical principles, and applications to verifying sequential, multi-threaded, and real-time software. The tutorial will be hands-on, with live demonstrations of using BMC tools for verifying sample programs written in C.

Categories and Subject Descriptors

D.2.4 [Software/Program Verification]: Model checking; F.3.1 [Specifying and Verifying and Reasoning about Programs]: Mechanical verification

Keywords

Model Checking; Verification; Bounded Model Checking; Software Verification; Real-Time Software

2. OUTLINE

The total duration of the tutorial is 3.5 hours. The topics to be covered (including estimated duration of each topic) are:

1. Overview of model checking (15 mins)
2. Overview of SAT/SMT solving (15 mins)
3. Bounded model checking of hardware – connecting model checking and SMT (20 mins)
4. Bounded model checking of single-threaded C code (50 mins)
   - Technical details (verification condition generation etc.)
   - Hands-on demonstration using the cbmc tool – http://www.cprover.org/cbmc
5. Bounded model checking of multi-threaded C code (30 mins)
   - Technical details (sequentialization for multi-threaded code)
   - Hands-on demonstration using the cbmc tool
6. Bounded model checking of periodic real-time software (50 mins)
   - Technical details (sequentialization for periodic programs)
   - Hands-on demonstration using the rekh tool – http://www.andrew.cmu.edu/~arieg/rekh
7. Summary, Ongoing and Future Work, Q&A (30 mins)

3. INTENDED AUDIENCE

This tutorial is aimed primarily at someone looking for a hands-on introduction to bounded model checking for single-threaded, multi-threaded, and periodic real-time software, e.g., someone with a background in software development, compilers, static analysis, software quality, programming languages etc. Knowledge about one or more of the following topics is desirable:

---

1. \(\text{http://www.satcompetition.org}\)
2. \(\text{http://www.smtlib.org}\)
1. Compiler internals (e.g., control flow graphs, static single assignment).
2. Propositional logic and satisfiability.
3. Familiarity with an imperative programming language (C preferred).
4. No prior knowledge about model checking, temporal logic etc. needed.

4. PRESENTER INFORMATION

Sagar Chaki is a senior Member of Technical Staff at the Software Engineering Institute at Carnegie Mellon University. He received a B.Tech in Computer Science & Engineering from the Indian Institute of Technology, Kharagpur in 1999, and a Ph.D. in Computer Science from Carnegie Mellon University in 2005. These days, he works mainly on model checking software for real-time and cyber-physical systems, but he is generally interested in rigorous and automated approaches for improving software quality. He has developed several automated software verification tools, including two model checkers for C programs, MAGIC and Copper.

5. ACKNOWLEDGMENT

Copyright 2013 ACM. This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESS OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material has been approved for public release and unlimited distribution. DM-0000626

6. REFERENCES


ABSTRACT
Recommender system aim at providing a personalized list of items ranked according to the preferences of the user, as such ranking methods are at the core of many recommendation algorithms. The topic of this tutorial focuses on the cutting-edge algorithmic development in the area of recommender systems. This tutorial will provide an in depth picture of the progress of ranking models in the field, summarizing the strengths and weaknesses of existing methods, and discussing open issues that could be promising for future research in the community. A qualitative and quantitative comparison between different models will be provided while we will also highlight recent developments in the areas of Reinforcement Learning.

Categories and Subject Descriptors
H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—Information Filtering, Retrieval Models

Keywords
Collaborative filtering, learning to rank, ranking, recommender systems

1. INTRODUCTION
Recommender systems aim to provide users with personalized items, which are typically ranked in a descending order of predicted relevance [1]. Learning the personalized recommendation list can be cast as a ranking problem. Naturally this cutting-edge research topic, Learning to Rank (LtR) [4], has already attracted a lot of attention in the Information Retrieval and Machine Learning communities. Recent contributions to collaborative filtering (CF) have exploited LtR techniques for improving the ranking of the top-N recommendations. In this tutorial, we present the key ideas of different categories of learning to rank approaches, and demonstrate examples that extend these ideas to specific CF methods. We also discuss a few open issues that remain challenging for future research in this direction.

2. OVERVIEW

2.1 Background
The tutorial briefly introduces the background of recommender systems and CF techniques. In particular, conventional CF methods target the rating prediction problem, such as the problem defined in Netflix Prize contest 1. However, we emphasize that the more important objective in recommender systems is ranking, or the top-N recommendation. We also review some conventional ranking methods for recommendation, such as item-based CF [8].

2.2 LtR for Recommender Systems
We introduce the concept of LtR in the area of information retrieval, and explain its usefulness for recommender systems based on the analogy between query-document search and user-item recommendation. Then, we review three types of LtR techniques, i.e., point-wise, pair-wise and list-wise LtR.

CF methods that learn a ranking model based on the preference scores of individual items can are considered point-wise ranking methods [3]. With pair-wise LtR, CF methods can be developed to take into account the preferences of each user to a pair of items. A typical example of CF with pair-wise LtR is Bayesian personalized ranking [7]. CF methods based on list-wise LtR model the list-wise preferences of each user to a list of items (usually the rated items). An important branch of CF methods in this category are designed to directly optimize the evaluation metrics, such as mean average precision (MAP), Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain (NDCG), which are usually list-wise ranking measures. Typical examples of those methods are CofiRank [12], ClMF [10] and TFMAP [9].

2.3 Implicit and Explicit Feedback
The core idea behind CF is that users whose past interests were similar will also share common interests in future. The user’s interest is inferred by the user interaction patterns with the items either explicitly or implicitly. In explicit feedback, users are asked/allowed to explicitly rate the items that have been purchased/consumed, using a pre-defined Likert scale (graded relevance), e.g., 1-5 stars in

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s). Copyright is held by the author/owner(s).

RecSys ’13, October 12–16, 2013, Hong Kong, China.
ACM 978-1-4503-2409-0/13/10.

1http://www.netflixprize.com/
Netflix movie recommendation site. Higher grade indicates higher preference/relevance of the item. In implicit feedback [2], users interact with items by downloading, purchasing, and their preferences are deduced from the interaction patterns. In this paper we propose a new model for the case of explicit/graded relevance data.

Various models for implicit feedback data use learning to rank [4] techniques to optimize binary relevance data ranking metrics. For example, several CF models [7, 9, 10] compute near optimal ranked lists with respect to the Area Under the Curve (AUC), Average Precision (AP) [5] and Reciprocal Rank [11] metrics. However, metrics that are defined to handle binary relevance data are not directly suitable for graded relevance data. Binary metrics, and CF methods that optimize for these metrics can be used on graded relevance data if it is converted to binary relevance by e.g., imposing a threshold (e.g., setting rating 4 as the threshold for the 1-5 scale so that items rated 4 and 5 are treated as relevant). This process has two major drawbacks: 1) we lose grading information within the rated items, e.g., items rated with a 5 are more relevant then items rated with a 4. This information is crucial in building precise models. 2) the choice of the thresholding relevance is arbitrary and will have an impact on the performance of different recommendation approaches.

2.4 Interactive Recommendation

In many recommendation domains the content is often dynamic and/or short lived such as video, news recommendation and computational advertising. In these domains it is often difficult to collect enough information in the form of clicks or ratings to recommend an item accurately using standard CF methods. Exploration/exploitation methods from reinforcement learning such as Multi-armed Bandits are particularly well suited for these domains. LTR techniques can be also used for Multi-armed Bandits [6].

2.5 Open Issues

While there has been significant progress in the development of novel ranking methods, one significant issue in CF methods and recommender systems in general is modeling the dependencies of the items in a recommendation list. Modeling the dependencies within a recommendation list can potentially produce more accurate ranking, increased diversity and novelty. Other open issues despite significant progress include the efficient modeling of context (e.g., location) and item content information in ranking models.

3. OBJECTIVES

The topic of this tutorial focuses on the cutting-edge algorithmic development in the area of recommender systems. This tutorial would bring a big picture of the progress of ranking models in the field, summarizing the strengths and weaknesses of existing methods, and discussing open issues that would be promising for the future research in the community. The tutorial is intended for researchers and practitioners in the area of recommender systems, especially those who are interested in recommendation algorithms.

4. ACKNOWLEDGEMENTS

This work is funded as part of a Marie Curie Intra European Fellowship for Career Development (IEF) award (CARS, PIEF-GA-2010-273739) held by Alexandros Karatzoglou.

5. REFERENCES