

Type-Based Security for Mobile Computing: Integrity, Secrecy and Liveness

Nobuko Yoshida, Department of Computing, Imperial College London

1 Background

Nowadays the use of mobile code is widespread throughout computing scenes, from intra-net applications in corporation to migrating code in active network to dynamic content in the world-wide web to telephony applications [20, 26, 32, 33]. One of the principal benefits of mobile code (as found in Java and CLR) is to allow *extensibility* [9, 31], where a piece of code migrates from a source node to a target node and gets linked to the run-time environment of the target, to serve its purposes. Such extensibility leads to rich repertoire of new functionalities, including dynamic, tightly-coupled use of remote computational resources not restricted by network bandwidth/latency, multi-media applications where real-time interaction with resources are essential, and incremental software/service addition/update. This open characteristic of mobile computation poses a new challenge in software safety: the infrastructure is required to manage security mechanisms by which pieces of code with different origins and functionalities, embodying different principals, can safely interact in the presence of mobility and dynamic linkage. This research proposal develops basic technologies to solve these security issues concerning mobile programs using well-defined mathematical models, namely typed mobile processes, and applies them to developing programming language disciplines. In the following, we first discuss the main issues this project tries to solve and how the project will achieve its aims.

1.1 Security Issues on Mobile Computation

Let us assume a simple e-commerce mobile agent, acting as a customer, which moves through different sites on the Internet in order to perform commercial transactions. The agent may be equipped with a buying list and orders on behalf of its owner as its script (program) and perform complex interactions with virtual shops. It may also change its behaviour depending on, e.g. prices of goods. Then there are three major issues arising during the interaction between agents (clients) and shops (hosts).

- the agent may jeopardise the *integrity* of the host; for example, the agent may delete or modify sensitive information such as the sales account of the shop.
- the agent may violate the *privacy*, or the *secrecy* of the host; for example, the agent may transfer credit card numbers stored at the host to the public channels.
- the agent may interfere with the *availability*, or the *liveness* [28] of services; for example, after an initial interaction, the agent may not return the acknowledgement, entering into an infinite loop. Then the host cannot proceed further (this is one form of the so-called *denial of service attacks*).

Note, dually, the agents may suffer from the same kinds of threats caused by malicious or erroneous hosts. This proposal aims to solve the above three central security issues concentrating on source code mobility, by the development of a general theory which is directly applicable to practical, real-life programming languages.

1.2 Type-Based Approach for Security and Current Technical Problems

In the 1990s, most new software was written in languages such as C, C++ and Java, which all feature varying degrees of static type checking. In current applications, the main property guaranteed by typing systems is still very simple type soundness: “well-typed programs do not go wrong”, e.g. do not apply integer if it is typed as string. Violations of type soundness constitute real security threats, and many researchers have shown type soundness of subset of Java can detect such threats, e.g. [8, 18]. However, technically speaking, three security issues, integrity, privacy of data and liveness, are not easily guaranteed by simple type safety; for example, in the current security architecture of Java and C#, the basic type checking is done before execution, but access control is done dynamically in a restricted and adhoc manner, whose effects are difficult for most programmers to predict and even to interpret [1, 5]. Against this background, one of the urgent issues which has not yet been satisfactorily addressed is *static, in particular, type-based verification of mobile code for properties beyond simple type safety*.

At the research level, static typing systems in sequential and functional programs have been used successfully for guaranteeing termination of programs, for controlling privileges and capabilities (e.g. [5, 24]), and for reasoning about information flow of programs (e.g. [7]). In this context, one recent successful research development is a commercial achievement on Proof Carrying Code (PCC) [6, 11, 23], which ensures the safety of mobile code statically. One important point is that this approach is crucially backed up by *semantics* and *types* — Hoare logics as a specification logic and the typed λ -calculus (a variant of Edinburgh Logical Framework (LF) [12]) for representing certificates of mobile code and for formally checking untrusted code. This framework opens a wide possibility to use a formal foundation to control code mobility by static checking. On the other hand, the current version of PCC [6, 11] cannot deal with concurrent or non-deterministic programs as transferable code. In fact, there is a lack of formal foundations of typing, logics and semantics to even express (not to speak of proving) desired security properties of mobile programs when they include *non-determinism*, *communication*, *concurrency* and *distribution*, in contrast to sequential programs. As another example, the resource preservation guaranteed by strong normalisation has been one of the main reasons for SwitchWare Project to develop their typed programming language for active networks

(PLAN) [25] on the basis of a simply typed λ -calculus. Jflow [19] designs a secrecy type discipline for a major subset of Java and studies its implementation involving possibly untrusted hosts [39]. Currently, incorporation of concurrency, communication and program distribution has not been considered in these languages, even though these elements are essential to modern software.

From these observations, we have the following questions:

- can we construct a static typing system beyond simple type safety, extending the accumulated theories of functional and sequential types to mobile processes?
- can we use types of processes to design secure mobile languages?
- can we apply theories of process calculi for the use of real applications in this domain?

This proposed project seeks to give a positive answer to the above questions by demonstrating the effectiveness of a general semantic theory of typed mobile processes in practice. The technical programme is built on my recent advances in access controls for mobility [13, 35, 38], secrecy [15, 16, 37] and type theory for liveness and linearity of communication [2, 3, 17, 36].

2 Project Aim

The main aim of the present project is to develop a general and uniform framework for type systems of mobile programs with respect to liveness, access control and secrecy preservation. Then we offer a language design discipline for safe mobile programs based on these typing systems. In order to achieve this goal, we tackle these issues at three different levels (base language, mobility and application) using three different formalisms/languages: the π -calculus [21], the higher-order (HO) π -calculus (which is an integration of the λ -calculus and the π -calculus [27, 35]), and a subset of Java [4, 8, 18]. The generality of the (HO) π -calculus plays an essential rôle; the resulting framework is, by instantiation, applicable to individual practical languages. Concretely, the project will focus on the following three activities.

(base language level) Establishing an integrated framework of the typing systems of the π -calculus which can fully abstractly embed various language primitives such as assignment, procedure call, polymorphism, controls and objects. The framework should in particular capture basic liveness property and is enhanced by secrecy.

(mobility level) Developing the basic typing systems of the higher-order π -calculus (HO π -calculus) by further extending the typing system for the HO π -calculus introduced in [35] in order to investigate more advanced access control and location primitives [13]. We then merge it to ensure secrecy and liveness based on the results of the base language level, obtaining a powerful typed meta-language for source code mobility.

(application level) Designing a subset of multi-threaded Java and its typing system which can ensure three security concerns, based on the typing systems developed in the mobility level. This track is carried out with Research Fellow. We prove its correctness (safety) via fully

abstract translation into the HO π -calculus. For the use in the framework of PCC [6, 11, 23], we also plan to construct automatic verification tools for the typing system adapting those studied in [10].

References

- [1] Abadi, M. and Fournet, C., Access Control based on Execution History, *NDSS'03*, California, 2003.
- [2] Berger, M., Honda, K. and Yoshida, N., Sequentiality and the π -Calculus, *TLCA01*, LNCS 2044, pp.29–45, Springer, 2001.
- [3] Berger, M., Honda, K. and Yoshida, N., Genericity and the π -Calculus, *FoSSaCs'03*, LNCS, Springer-Verlag, 2003.
- [4] Bierman, G.M., Parkinson, M.J and Pitts, A.M., *MJ: an imperative core calculus for Java and Java with effects*, UCAM-CL-TR-563, Cambridge, 2003.
- [5] Fournet, C. and Gordon, A.D., Stack Inspection: theory and variants, *POPL*, ACM, 2002.
- [6] Colby, C. et al., A Certifying Compiler for Java, *Proc. PLDI00*, ACM SIGPLAN, 2000.
- [7] Denning, D. and Denning, P., Certification of programs for secure information flow. *Communication of ACM*, ACM, 20:504–513, 1997.
- [8] SLURP Project, Department of Computing, Imperial College, London, <http://binarylord.com/slurp/>.
- [9] Grimm, R. and Bershad, B., Separating Access Control Policy, Enforcement, and Functionality in Extensible Systems, *ACM Tra. on Comp. Sys.*, 19(1), Feb., 36–70, 2001.
- [10] Gay, S., A Framework for the Formalisation of Pi-Calculus Type Systems in Isabelle/HOL, *Proc. TPHOLs01*, Springer, LNCS, 2001.
- [11] Fox Project, <http://www-2.cs.cmu.edu/~fox/pcc.html>
- [12] Harper, R., Honsell, F., and Plotkin, G., A framework for defining logics, *Journal of the Association for Computing Machinery* 40(1), 143–184, 1993.
- [13] Hennessy, M., Rathke, J. and Yoshida, N., SafeDpi: a language for controlling mobile code, To appear in *FoSSaCs'04*, LNCS, Springer, 2004.
- [14] Honda, K. and Tokoro, M., An Object Calculus for Asynchronous Communication. *ECOOP'91*, LNCS 512, pp.133–147, Springer 1991.
- [15] Honda, K., Vasconcelos, V. and Yoshida, N., Secure Information Flow as Typed Process Behaviour, *ESOP'00*, LNCS 1782, pp.180–199, Springer, 2000.
- [16] Honda, K. and Yoshida, N., A Uniform Type Structure for Secure Information Flow, *POPL'02*, pp.81–92, ACM SIGACT-SIGPLAN, ACM Press, 2002.
- [17] Honda, K., Berger, M. and Yoshida, N., Controls in the π -Calculus, *Proc. CW'04*, ACM Press, 2004.
- [18] Igarashi, A., Pierce, B. and Wadler, P., Featherweight Java: A Minimal Core Calculus for Java and GJ. *TOPLAS*, 23(3):396-450, May 2001.
- [19] JFlow Project, <http://www.cs.cornell.edu/jif/>.
- [20] Knapik, M. and Johnson, J., *Developing Intelligent Agents for Distributed Systems*, McGraw Hill, 1998.
- [21] Milner, R., Parrow, J.G. and Walker, D.J., A Calculus of Mobile Processes. *Information and Computation*, 100(1), pp.1–77, 1992.
- [22] Sabelfeld, A. and Meyers, A., Language-Based Information Flow Security, *IEEE Journal on Selected Areas in Communications*, 21(1), January 2003.
- [23] Necula, G., Proof-carrying code. *POPL'97*, pp.106–119, ACM, 1997.
- [24] Pottier, F., Skalka, C. and Smith, S., A systematic approach to access control. *ESOP01*, LNCS 30–45, Springer, 2001.
- [25] PLAN: A Packet Language for Active Networks, SwitchWare Project, <http://www.cis.upenn.edu/~switchware/>.
- [26] RIFML, the Reactive Intelligence Framework, Enigmatec Corporation Ltd, <http://www.enigmatec.net/>.
- [27] Sangiorgi, D., *Expressing Mobility in Process Algebras: First Order and Higher Order Paradigms*. Ph.D. Thesis, Univ. of Edinburgh, 1992.
- [28] Schneier, F., *On Concurrent Programming*, Springer, 1997.
- [29] Smith, G. and Volpano, D., Secure information flow in a multi-threaded imperative language, pp.355–364, *POPL'98*, ACM, 1998.
- [30] VerifiCard, a European Project for Smart Card Verification, <http://www.verificard.org>.
- [31] Wallach, D.S, et al., Extensible security architectures for Java. *SOSP*, 116–128, IEEE, 1997.
- [32] XLANG, Web services for business process design, http://www.getdotnet.com/team/xml_wssspecs/xlang-c. (c) Microsoft 2001.
- [33] Web Services Choreography Working Group, <http://www.w3.org/2002/ws/chor/>.

- [34] Yoshida, N. and Hennessy, M., Subtyping and Locality in Distributed Higher Order Processes. *Proc. CONCUR'99*, LNCS 1664, pp.557–573, Springer-Verlag, 1999.
- [35] Yoshida, N. and Hennessy, M., Assigning Types to Processes, *Info. & Comp.*, 174(2), pp. 143-179, Academic Press, 2002.
- [36] Yoshida, N., Berger, M. and Honda, K., Strong Normalisation in the π -Calculus, *Proc. LICS '01*, pp.311-322, IEEE, 2001. The full version: to appear in *Journal of Information and Computation*.
- [37] Yoshida, N., Berger, M. and Honda, K., Linearity and Bisimulation, *Proc. FoSSaCs'02*, LNCS 2303, pp.417–433, Springer-Verlag, France, 2002.
- [38] Yoshida, N., Channel Dependency Types for Higher-Order Mobile Processes, *POPL '04, Conference Record of the 31st Annual Symposium on Principles of Programming Languages*, ACM SIGACT-SIGPLAN, ACM Press, 2004.
- [39] Zdancewic, S. Zheng, L., Nystrom N., Myers, A., Untrusted Hosts and Confidentiality: Secure Program Partitioning, *SOSP01*, ACM, 2001.