1997

Provably-Secure Programming Languages for Remote Evaluation

Volpano, Dennis

Monterey, California. Naval Postgraduate School

Provably-Secure Programming Languages for Remote Evaluation

Dennis Volpano
Naval Postgraduate School, Monterey

Remote evaluation and dynamically-extensible systems pose serious safety and security risks. Programming language design has a major role in overcoming some of these risks. Important research areas include designing suitable languages for remote evaluation, identifying appropriate security and safety properties for them, and developing provably-sound logics for reasoning about the properties in the context of separate compilation and dynamic linking.

Recently, there has been phenomenal growth in the use of the Internet, driven in part by HTTP servers that make up the World Wide Web and new programming languages like Java [5]. This has rekindled interest in an area that Stamos and Gifford termed *remote evaluation* [12]. In this paradigm, a client sends a procedure to a server to execute on its behalf. The server executes the procedure and returns the results to the client. This differs from remote procedure call (RPC) and other forms of gateway programming which are examples of server-side programming, not remote evaluation.

Various languages have been proposed to support the paradigm. Among them are distributed dialects of Scheme [4; 7], Tcl (Safe-Tcl) which grew out of active messaging [2], Obliq [3] and General Magic's Telescript(tm) [14]. Sun's Java is also aimed at remote evaluation but from a slightly different perspective. With Java, a procedure in byte-code form, called an Applet, is downloaded from a Web server and executed within a browser on the client's machine.

It has long been widely known that there are serious security risks associated with remote evaluation. By security, we mean server privacy, integrity and availability. Remote procedures that execute with server privileges and have access to server resources can compromise security. This is a real threat to the future of Internet computing and dynamic extensibility in OS kernels and active networks.

Though attention has been given to security issues in this setting, they have not been treated with anywhere near the rigor found in other areas of security such as encryption. Languages like Obliq, Netscape's JavaScript and Java have not been carefully designed from a security point of view. As new holes are discovered, they are patched, but it is not clear that this process will ever converge. One can never be sure that executing programs in these languages will not compromise security in some way. As a result, browsers tend to be quite paranoid when it comes to executing remote code. They implement rather restrictive security policies that prevent all but very anemic applications from being executed remotely.

1. SECURITY PROPERTIES FOR PROGRAMMING LANGUAGES

Many security issues in remote evaluation can actually be addressed through careful and systematic programming language design. What is needed is a provably-secure programming language, one whose design is guided by a need to preserve some set of explicitly-stated security properties. But what kinds of security properties do we want programs to have? First, we might expect the language to be, in some sense, *safe*. That is, it should have features that promote robust code and avoid accidents, unlike C. We want a precise characterization of how a well-typed program can behave when executed. New formulations of type soundness are needed for imperative programming languages that specify all possible errors that can cause well-typed programs to abort according to the semantics. Traditional type soundness arguments merely rule out well-typed programs from evaluating to a special type-error value. These new formulations will force one to identify, in the semantics, various points where program execution should abort, for example, when attempting to dereference a dangling pointer. The idea is that a safe implementation of the language would then be required to detect these points. An open question is what features can a programming language have that allow it to be implemented safely *and* efficiently?

For other security properties, we can look to *security models* for information flow in multi-level systems [9; 10; 11]. Various models such as, Noninterference [6], Separability [10] and Restrictiveness [8] have been proposed. They are basically properties of multi-level systems that say high-level system inputs do not interfere with low-level system outputs. Each security model offers a different notion of security. The Noninterference model, for example, addresses protection for program inputs only and is not a property of nondeterministic programs. It may be too weak in some cases. Consider programs that generate cryptographic keys, for example. They are expected to convert low-level input seeds into high-level output keys. Noninterference would not be concerned with whether these keys wound up being low-level outputs. Separability, on the other hand, is a stronger notion of security and is a property of nondeterministic programs. However, its weakness is that it prohibits upward information flow from low-level inputs to high-level outputs, making it unsuitable for some applications. An important research direction is to identify an appropriate set of security properties for remote evaluation languages.

2. PROOF SYSTEMS FOR SECURITY PROPERTIES

It should be possible to enforce a set of desired security properties through a proof system for the language. Of course, whether the properties are enforced by the proof system must be shown through a soundness theorem which is stated with respect to the language's semantics. The theorem guarantees that all programs that have proofs in the system have the desired security property. For instance, a proof system has been designed to enforce Noninterference in a deterministic, block-structured language and has been proved sound [13]. The proof system is formulated as a type system so that well-typed programs have the Noninterference property. Depending on the security model and programming language, getting a provable formulation of soundness can be tricky. For example, Banâtre et al. give an information flow logic for a nondeterministic language [1]. However, in order

for their formulation of soundness (Proposition 1, pg. 58) to be true, the flow logic must be changed [13]. There are also algorithmic issues surrounding such proof systems. Is a particular proof system decidable? If so, can it be decided efficiently?

The future of Internet computing and extensible systems holds great promise. A key to its success is security, and provably-secure programming language design will have a major role.

REFERENCES

[1] Banâtre, J., Bryce, C. and Le Métayer, D., Compile-time Detection of Information Flow in Sequential Programs, *Proc. 3rd European Symposium on Research in Computer Security*, pp. 55–73, 1994.

[2] Borenstein, N., Email with a Mind of its Own: The Safe-Tcl Language for Enabled Mail, Available at `ftp://ics.uci.edu/mrose/safe-tcl/safe-tcl.tar.Z`, 1994.

[3] Cardelli, L. A Language with Distributed Scope, *Proc. 22nd ACM Symposium on Principles of Programming Languages*, pp. 286-297, 1995.

[4] Cejtin, H. Jagannathan, S. and Kelsey, R., Higher-order Distributed Objects, *ACM Trans. on Programming Languages and Systems*, 17(5), pp. 704–739, 1995.

[5] Flanagan, D., Java in a Nutshell, O'Reilly and Associates, Inc.

[6] Goguen, J. and Meseguer, J., Security Policies and Security Models, *Proc. 1982 IEEE Symposium on Research in Security and Privacy*, pp. 11-20, 1982.

[7] Halls, D., Bates, J. and Bacon, J., Flexible Distributed Programming using Mobile Code, *Proceedings of the 1996 SIGOPS European Workshop on Systems Support for Worldwide Applications*, Connemara, Ireland, September 1996, Available at `http://mosquitonet.stanford.edu/sigops96/papers`.

[8] McCullough, D., Specifications for Multi-level Security and a Hook-Up Property, *Proc. 1987 IEEE Symposium on Research in Security and Privacy*, pp. 161-166, 1987.

[9] McLean, J., The Specification and Modeling of Computer Security, *IEEE Computer*, 23(1), pp. 9-16, 1990.

[10] McLean, J., Security Models and Information Flow, *Proc. 1990 IEEE Symposium on Research in Security and Privacy*, pp. 180-187, 1990.

[11] McLean, J., A General Theory of Composition for Trace Sets Closed Under Selective Interleaving Functions, *Proc. 1994 IEEE Symposium on Research in Security and Privacy*, pp. 79-93, 1994.

[12] Stamos, J. and Gifford, D., Remote Evaluation, *ACM Trans. on Programming Languages and Systems*, 12(4), pp. 537–565, 1990.

[13] Volpano, D., Smith, G. and Irvine, C., A Sound Type System for Secure Flow Analysis, *Journal of Computer Security*, 4(3), pp. 1–21, 1996.

[14] White, J., Telescript Technology: The Foundation for the Electronic Marketplace, Technical Report, General Magic, Inc., 1994.