Programming Language Semantics and Semantic Web Security

Dr. Kevin W. Hamlen
University of Texas at Dallas

Abstract.

Programming language-based security research, such as Proof-Carrying Code and In-lined Reference Monitoring, and Semantic Web security research have heretofore been largely disparate lines of inquiry. However, the rise of web scripting languages as an essential component of interactive web content has created an emerging need to bridge these two technologies. Without advances in this intersection of disciplines, scripting-based media will continue to constitute a conspicuous and dangerous gap in the development of a comprehensively secure semantic web.

The Semantic Web, first envisioned in 2001 by Sir Tim Berners-Lee [1], anticipates an evolution of the World Wide Web from a mere information rendering and storage medium to one that expresses the meaning of information in a machine-parseable way. Such advances open exciting new possibilities for autonomous knowledge discovery, inference, and reasoning, but can also facilitate information security attacks that exploit these new functionalities. For example, a semantic web in which autonomous agents can easily collect personal data (e.g., pet names and their owners) could be leveraged by an attacker to learn the answers to security questions commonly used by e-banking authentication systems. Such threats have led to a growing body of work on semantic web security issues, such as the application of semantic web specification languages to the security policy specification problem (c.f., [2]).

While researchers have been making steady progress over the last eight years toward realizing the Semantic Web vision, another trend has arisen in e-commerce web design: the increasing ubiquity of web scripting. Scripting languages such as JavaScript, ActionScript, and Active Server Pages (ASP) have become commonplace in rendering web page content, accepting web user input, and dynamically interacting with the user and third parties. Although hard statistics on scripting language usage have not been collected to our knowledge, it seems evident that these trends are increasing. In fact, a cursory examination of web page source codes reveals that many modern e-commerce pages now consist solely of a small HTML stub that houses a script that contains all of the significant page content.

Given these trends, we consider the intersection of web script security and semantic web security to be a significant source of emerging information security challenges for the next decade. These challenges include the development of:

1) **Script-aware, ontologically-based policy specification languages** that connect information integrity, confidentiality, and trust assertions with semantic information and script behavior requirements,
2) **Script policy negotiation protocols** that allow dynamic selection and/or generation of web scripts satisfying a negotiated information security policy, and

3) **Automated script verification algorithms** that allow script recipients to safely execute web scripts obtained from untrusted script producers.

Last year we developed SPoX [3], a purely declarative, XML-based policy specification language for Java bytecode applets. SPoX policies describe the set of allowable script behaviors by constraining the history of security-relevant events (usually accesses of system services) they may exhibit at runtime. These policies are enforced by web clients rather than servers by automatically analyzing and rewriting untrusted Java applets to inject runtime security checks. If the original applet is already policy-adherent, these inserted checks have no observable effect at runtime; but if the script would have violated the policy, the transformed script is prevented from doing so (e.g., by prematurely terminating it with a security warning).

While SPoX is an extremely powerful and expressive specification language for web scripting security, it is still non-trivial to write correct specifications that accurately model a desired high-level policy. Policy authoring therefore typically requires the assistance of a scripting language expert. Semantic web technologies seem to hold great promise for alleviating this burden in that they expose the critical ontological information about inputs and outputs to/from web scripts that, when paired with a more general information flow specification, give rise to a suitable SPoX specification. In future work we would like to investigate the feasibility of enforcing end-to-end information assurance policies by relating SPoX policies to semantic web policy languages in this way.

While programming language-based security research such as that outlined above and semantic web security research have heretofore been largely disparate, the rise of scripting languages as an essential web fixture makes it vital to discover and analyze security enforcement methodologies that bridge these two technologies. Without such advances, scripting-based information rendering will continue to constitute a conspicuous and dangerous gap in the development of a comprehensively secure semantic web.

