SECURE SEMANTIC COMPUTING

KEVIN W. HAMLEN* and BHAVANI THURAISINGHAM†

Computer Science Department
The University of Texas at Dallas
Richardson, TX 75080, USA
**hamlen@utdallas.edu*
†*bhavani.thuraisingham@utdallas.edu*

This paper explores the integration of semantic computing technologies with security technologies. Past and current research on the application of semantic web technologies for policy management and inference control, the application of data mining technologies for intrusion and malware detection, and programming language-based approaches to mobile code certification and data confidentiality enforcement are discussed.

Keywords: semantic web; security; inference control; privacy; information flow controls

1. Introduction

The explosive growth of the Web since its advent in the mid-1990s has generated increasing demand for tools and algorithms that can effectively manage data, information, and knowledge on a mass scale. There is now so much data on the Web that managing it with conventional tools is becoming almost impossible. For example, Google continually invents and deploys increasingly distributed, decentralized technologies to maintain its indexing system, which now stores and serves tens of petabytes of data with updates processed in parallel by thousands of machines per day worldwide [1]. Google and most other web search engines are primarily limited to syntax-based queries, with search results based on keyword and phrase lookups. However, these syntactic searches place heavy reliance upon a human user, who must parse, filter, and analyze the search results to infer semantic properties of the data. This limits their usefulness for automating complex tasks that require a machine agent to reliably obtain answers to many thousands of queries without user assistance.

To mine and manage these more sophisticated semantic properties, Tim Berners Lee's Semantic Web expresses semantic properties via machine-understandable web pages [2]. This allows data mining techniques to more effectively analyze the information and reliably extract information not available via syntactic search. Semantic web technologies have given rise to a broad array of semantic computing technologies, which facilitate automated reasoning about semantic properties at all levels of computing, including the operating system, network, database, programming language, and applications levels. This in turn facilitates the development of

intelligence systems and networks, which can use the information to automatically discover otherwise hidden patterns and relationships within the data. The ability of semantic computing to increase the availability of these data relationships has introduced both new security opportunities and new security threats. Study and development of secure semantic computing technologies is therefore an important emerging category of security research.

Our work has focused on three aspects of secure semantic computing technologies: First, we focus on secure semantic technologies for designing and developing intelligent web information management systems that store and process vast quantities of data. Data has become a critical resource in many organizations for which efficient and reliable data access, sharing, and information extraction is essential to organization operation. Hence, it is necessary to protect data and information from both unauthorized access and malicious corruption. Our main focus here is on inference control and policy specifications with semantic web technologies. Second, we examine data mining technologies for cyber-security applications. In particular, we discuss data mining for intrusion detection and malicious code detection. We also discuss privacy-preserving data mining technologies. Third, we consider the importance of programming language semantics for meta-reasoning in semantic webs—that is, the ability to automatically reason about the machine agents that process and purvey semantic web data.

2. Semantic Technologies, Policy Management, and Inference Control

Semantic technologies are playing a major role in policy management and inference control. In the 1990s, technologies such as Datalog were used to specify and reason about policies. A hallmark of this work was the development of inference controllers that protect relational database systems from confidentiality attacks that infer secrets from non-confidential data. Datalog-based reasoners would reason about the past history of query responses released to the user and determine which further information is releasable without violating the confidentiality policy [3]. An example of such an inference controller is illustrated on the left of Figure 1.

More recently, semantic web technologies are being applied to formally specify policies for data management as well as information sharing. The significant advantage of Semantic Web technologies is their representational and reasoning power. This representational power can be leveraged to represent security policies about the data along with the data itself. For example, XML-based languages such as *eXtensible Access Control Markup Language* (XACML) have been used to specify various types of security policies. Researchers have proposed extensions for XACML for finer-grained access control. Furthermore, languages such as Resource Description Framework (RDF) are also being explored for representing the policies.

Reasoning engines such as Jena [4] and Pellet [5] are also being explored for representing and reasoning about Semantic Web-based policy specifications and

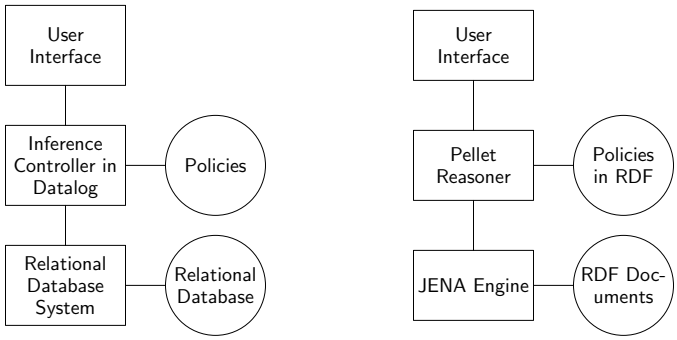

Fig. 1. An inference controller (left) and its analog for semantic web (right)

determining whether there are security violations through inference. For example, Jena manages RDF graphs, and Pellet reasons with RDF graphs. We have applied these technologies to develop a semantic web-based inference controller [6], depicted on the right of Figure 1. Semantic web technologies have also been applied for policy management for organizational data sharing [7]. In this case, each organization specifies its policies in XML, RDF, or Web Ontology Language (OWL). When data is shared among organizations, the policies are applied and only information for which the requesting principals are authorized is shared. Policy-based information sharing is illustrated in Figure 2.

Inference control is important not only for reasoning at the granularity of separate, security-relevant data objects and how they relate, but also for reasoning about the internal structure and content of individual data objects. For example, research on securing XML content and its schemas is currently investigating how to control access to various portions of sensitive text documents for reading, browsing, and modifications. To illustrate, consider the sentence, "President Obama is traveling to Iraq on April 20, 2011." A coarse-grained policy language might be forced

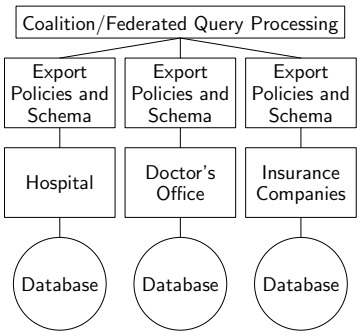

Fig. 2. Policy-based information sharing

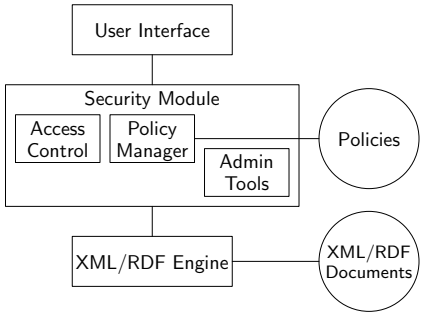

Fig. 3. Access control for semantic web-based documents

to permit or deny access to this sentence in its entirety, whereas a finer-grained, semantic-based policy language could permit access to alternate sentences that convey a semantic subset of the information—e.g., "President Obama is traveling to Iraq" or "President Obama is traveling on April 20, 2011." An inference control system could additionally deny access to one of these two latter sentences once the other has been accessed in order to prohibit inference of the complete sentence. Such security frameworks enforce policies on RDF (and the XML that underlies it) based on semantic interpretations of the data. Figure 3 illustrates access control for documents based on semantic web technologies such as XML [8].

Secure semantic technologies are also being applied for ontology alignment. Massive data repositories shared by many organizations, divisions, and users inevitably develop conflicting semantic structures and classifications for common data. For example, the marketing department might classify customer last name data under the concept "Surname," while the customer support department might classify the same information as "Family name" in troubleshooting tickets. Ontology alignments identify matching concepts across ontologies. This is very important for homeland security applications where such alignments often correspond to critical security information. For example, an ontology alignment inference algorithm might determine that John A. Smith and John Smith are actually aliases for the same person with high probability.

In our research, we are investigating an approach to this problem based on the path difference among concepts in the ontologies combined with an approximate privacy-preserving matching technique [9]. The approach is illustrated in Figure 4. Path differences are computed by comparing the path leading from each concept to the root of its ontology. The more similar these paths are, the more semantically similar the concepts. By compactly encoding these paths as strings (e.g., by adopting numeric encodings of concepts), we can discover and replace corresponding values in the original records. We can then match these modified records by using an approximate privacy-preserving matching technique that privately computes the distances between pairs of records and returns only record pairs whose distance

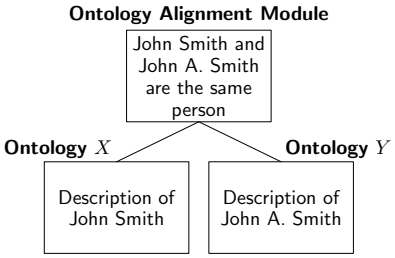

Fig. 4. Confidentiality- and privacy-preserving ontology alignment

remains below a certain threshold.

Privacy-preservation is a vital consideration for secure ontology alignment because the alignment introduces new data relationships that, if disclosed, might violate the system's confidentiality policy directly or lead to confidentiality-violating inferences. For example, if the fact that John Smith is a CIA officer is classified as Top-secret, and aligning the ontologies reveals that John A. Smith and John Smith are the same person, then previously public information about John A. Smith may need to be reclassified, or the alignment itself may need to be classified, to preserve the secret.

This example also illustrates that secure ontology alignment can offer rare and valuable opportunities to automatically detect and correct policy specification errors through the detection of policy conflicts. Data security policy specifications for massive repositories are often highly complex and dynamic, and therefore a potentially significant source of system vulnerabilities. A flaw in the specification can lead to catastrophic data integrity violations, confidentiality violations, or both. When separate divisions or organizations formulate separate data security policy specifications for differing ontologies over common data, an alignment of the ontologies can reveal conflicts that should be conservatively reconciled and brought to the attention of administrators. Extending the example above, if the CIA has classified John Smith's affiliations as Top-secret and the FBI has classified John A. Smith's affiliations as Public, but ontology alignment infers that John Smith and John A. Smith are the same person, then the system can conservatively reclassify John A. Smith's affiliations as Top-secret (i.e., the join of Top-secret and Public in the security lattice) and bring the conflict to the attention of the authorized administrators of both ontologies.

3. Data Mining, Cyber-security, and Privacy

Data mining technologies can leverage the semantic policy specification and data representation techniques summarized in Section 2 to extract previously unknown information from large data repositories. Historically, data mining has been applied to unstructured or unorganized data sets as a basis for inferring the sorts of semantic information that is already explicit in semantic webs and other semantic

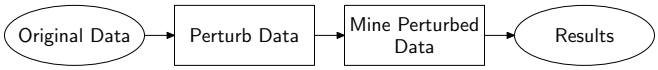

Fig. 5. Privacy-preserving data mining

computing frameworks (cf., [10]). Applying data mining to domains in which these semantics are readily available avoids this problem and can therefore leverage the explicit semantic information to infer more subtle data classifications, clusters, associations, and anomalies that are not reliably inferable by traditional data mining in the absence of semantic computing. Retargeting traditional data mining technologies, such as machine learning, statistical analysis, neural networks, association rule mining, decision trees, and genetic algorithms, to semantic computing domains is therefore an important area of converging research for both fields.

Our past work has applied data mining techniques for both national security and cyber-security applications. In the case of national security, a major goal is to gather data about terrorist behavior, analyze the data, and extract nuggets useful for predicting attacks and conducting counter-terrorism activities. One significant challenge for this research has been the development of approaches that provably protect the privacy of non-terrorists without sacrificing the precision and power of the algorithms for anticipating attacks. Collecting large amounts of information about many individuals generally improves predictive power, but also increases the danger of privacy violations in the form of unauthorized associations or direct leaks of private data. Therefore, researchers are examining ways to carry out data mining and at the same time ensure privacy. This area has come to be known as *privacy-preserving data mining* [11].

Figure 5 illustrates a perturbation approach to privacy-preserving data mining that introduces noise (i.e., random values) into the data in such a way that the perturbed data is difficult or impossible to mine for confidential data about individuals, yet higher-level associations and classifications are preserved and can therefore continue to be reliably mined by analysis tools. Alternatively, *multi-party computation* approaches allow data mining algorithms to be implemented in a distributed fashion by multiple mutually-distrusting organizations. Each organization verifiably computes its portion of the classification algorithm without divulging its private data to any of the others. The results of the distributed computations can then be combined to reveal the results of the analysis without risking disclosure of the private data to which the analysis was applied.

In the case of cyber-security, our prior work has implemented intrusion detection systems that mine data access logs for anomalies indicative of cyber-attacks [12]. Automated detection of previously unseen malware (i.e., zero-day attacks) is another important emerging application of data mining research that we have applied for cyber-defense [13]. Figure 6 illustrates data mining approaches to semantics-based intrusion detection.

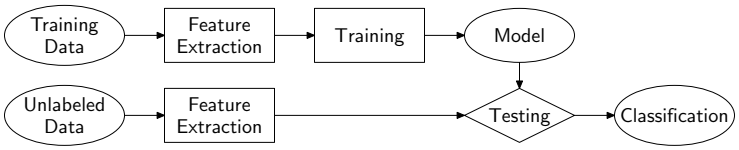

Fig. 6. Data mining for cyber-security

4. Semantic Computing and Programming Language Security

Scripts, web services, mobile bytecode applets, and other programmatic elements now pervade a large portion of the Web and other large information systems. For example, most e-commerce web sites use JavaScript code to dynamically organize and render product information for users, Flash applets to display web advertisements, Java applets to access customer information and inventory data from web services atop remote SQL databases, and a variety of other code to facilitate secure payment processing. Relatively little of the content exists as plaintext HTML or XML. These programmatic technologies are essential for scalably supporting dynamic content, on-demand data search and filtering, and data presentations that support user interaction. Reasoning effectively about security properties of large information systems therefore demands analyses that couple strong data semantic analyses with strong program semantic analyses.

Like inference control for data security, software security policies are temporal—each event's permissibility potentially depends on the history of previous events observed by the system. However, unlike inference control, which must anticipate all possible future behavior of human agents, the binary text of an untrusted program provides a rich field of semantic information that reduces the space of possible program behaviors that must be considered. This has led to a growing body of language-based security research that leverages program semantic information to enforce powerful, end-to-end security policies that consider both the data and code elements of complex systems [14].

In recent years, language-based security research has been dominated by work on enforcement mechanisms for two important classes of policies: *safety policies* and *information flow policies*.

4.1. *Safety Policy Enforcement and In-lined Reference Monitors*

Safety policies essentially say that some "bad event" must not happen. For example, to prevent unauthorized disclosure of private tax information, a tax calculator applet might be subjected to a policy that prohibits it from sending any messages to third-party advertisers once it has read the private information. Safety policies can be formalized as *security automata* [15], such as the one depicted in Figure 7. The edges of the automaton are labeled with predicates that match security-relevant events. The automaton thereby defines a language of permissible event sequences

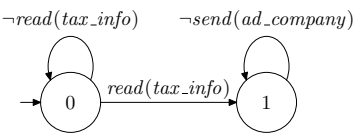

Fig. 7. A security automaton that prohibits sending private tax information to a third-party ad company

that programs may exhibit. In this case, the "bad event" is a send operation in state 1, for which there is no outgoing edge. Our past work has developed the first XML-based policy specification language for safety policies that constrain mobile Java bytecode [16]. Each specification denotes a security automaton in which edges are labeled by aspect-oriented *pointcuts*, which are expressions that denote sets of security-relevant program operations and their arguments.

Reference monitors enforce safety policies by monitoring untrusted programs as they run, simulating the security automaton and halting the program to prevent impending violations. However, reference monitors implemented at the OS level can be prohibitively inflexible in the sense that updating the monitor to enforce a new policy requires modifying the OS, which is difficult even when the OS source code is available, and impossible when it is purely binary code distributed over a network of mutually distrusting principals. This is a common scenario in mobile code frameworks. Language-based approaches to safety policy enforcement have therefore introduced the idea of automatically in-lining the code of the reference monitor directly into the binary code of untrusted programs. This yields self-monitoring mobile code whose enforcement mechanism is carried around with the mobile code itself. The in-lining process essentially identifies all binary operations that potentially change the security automaton's state, and instruments them with guard code that tracks the automaton state at runtime. If the guard code detects an impending violation at runtime, it takes corrective action, such as by self-terminating the process or rolling back to a policy-adherent state. We have successfully developed in-lined reference monitoring systems for .NET bytecode [17], Java bytecode [18], ActionScript Flash [19], and even x86 native code [20].

In-lined reference monitors (IRMs) have the additional benefit that they are amenable to formal verification. Such certified IRMs allow the in-lining process to be shifted to an untrusted third party, with the resulting self-monitoring code independently verified by the code-recipient. A certifying IRM framework is illustrated in Figure 8. Our work on .NET bytecode IRMs introduced a verification mechanism that formalized policy adherence as type-safety [17], permitting a simple client-side type-checker to independently verify that third-party in-liners yield self-monitoring code that is provably policy adherent on all possible runs. Our later work on ActionScript Flash IRMs for web ad security has introduced heavier-weight but stronger IRM verification systems based on model-checking [19].

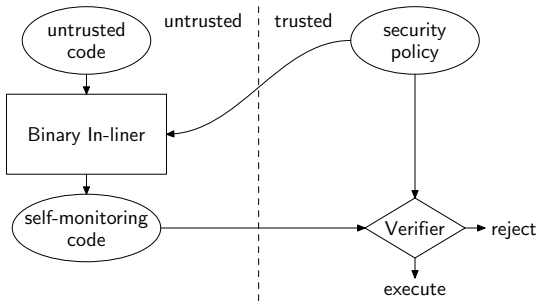

Fig. 8. A certifying in-lined reference monitoring framework

4.2. *Information Flow Control*

Information flow policies constrain the runtime flow of data from private sources to public sinks rather than constraining the program's history of observable operations. This is elegantly expressible as a static code semantics that augments a traditional type system with a lattice of classification labels [21]. For example, the programmer might annotate high-confidentiality integer variables with type int_{high} and low-confidentiality integer variables with type int_{low}. To protect against explicit information leaks, a type-checker can verify at compile-time that there are no explicit or implicit dataflows from variables or expressions of type τ_{high} to those of type τ_{low}. The Java Information Flow (JIF) system implements information flow controls via static type-checking for Java programs [22].

Two of the most significant long-term challenges for language-based information flow control research concern *covert channels* and *robust declassification*. Covert channels allow a passive attacker to infer confidential information from program outputs not intended for information communication, such as program timing and power consumption rates. For example, *differential power analysis* (DPA) [23] has been used to extract cryptographic keys from smart cards even in the absence of explicit or implicit, confidentiality-violating flows. Such power channels are extremely difficult to model accurately because they concern hardware details that can vary at the manufacturing level.

Robust declassification addresses the reality that most secure systems must intentionally release confidential information under certain conditions in order to perform their functions. A classic example is a password-checker, which divulges one bit of information about the confidential password to the attacker every time it rejects an incorrect guess. An information flow control system that has no escape mechanism by which a programmer can safely declassify information is too restrictive to be useful for most realistic applications. However, it is precisely the places where sanctioned declassification takes place that confidentiality vulnerabilities are most likely to arise. Therefore, an important ongoing area of study involves inventing technologies that can quantify and warn the programmer about excessive

information leaks at declassification sites.

5. Summary and Directions

Semantics-based computing is essential for supporting intelligent, autonomous agents and formal verification systems that reason about data and system properties at all levels, from operating systems to applications to the data they manipulate. However, the increasing adoption of semantic computing technologies introduces both new security challenges and new security opportunities. Challenges arise from the increasing availability of deep semantic information about confidential data. Inference control and data mining technologies can exploit this information to implement new forms of confidentiality attacks, but the same technologies form the basis for the most powerful defenses against these attacks. Similarly, language-based security technologies, such as certifying in-lined reference monitors and static, type-based information flow controls are important for reasoning about the programmatic components of the semantic web and other semantic computing frameworks.

References

[1] D. Peng and F. Dabek, Large-scale incremental processing using distributed transactions and notifications, in *Proc. USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2010.

[2] T. Berners-Lee, J. Hendler, and O. Lassila, The semantic web, *Scientific American* (May 2001) 35–43.

[3] B. M. Thuraisingham, W. Ford, M. Collins, and J. O'Keeffe, Design and implementation of a database inference controller, *Data & Knowledge Engineering* **11**(3) (1993) 271–.

[4] J. J. Carroll, D. Reynolds, I. Dickinson, A. Seaborne, C. Dollin, and K. Wilkinson, Jena: Implementing the semantic web recommendations, in *Proc. 13th International World Wide Web Conference on Alternate Track Papers & Posters (WWW)*, 2004, pp. 74–83.

[5] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, Pellet: A practical OWL-DL reasoner, *Web Semantics: Science, Services and Agents on the World Wide Web* **5**(2) (2007) 51–53.

[6] T. Grandison, *Data Provenance and Access Control with Semantic Web*, Ph.D. thesis, The University of Texas at Dallas, Richardson, Texas (August 2011).

[7] D. Harris, L. Khan, R. Paul, and B. M. Thuraisingham, Standards for secure data sharing across organizations, *Computer Standards & Interfaces* **29**(1) (2007) 86–96.

[8] E. Bertino, S. Castano, E. Ferrari, and M. Mesiti, Protection and administration of XML data sources, *Data & Knowledge Engineering* **43**(3) (2002) 237–260.

[9] H. Chen, AI and security informatics, *IEEE Intelligent Systems* **25**(5) (2010) 82–90.

[10] B. Thuraisingham, *Data Mining: Technologies, Techniques, Tools, and Trends* (CRC Press, 1998).

[11] R. Agrawal and R. Srikant, Privacy-preserving data mining, in *Proc. ACM SIGMOD International Conference on Management of Data*, 2000, pp. 439–450.

[12] M. M. Masud, T. Al-khateeb, L. Khan, B. Thuraisingham, and K. W. Hamlen, Flow-based identification of botnet traffic by mining multiple log files, in *Proc. 1st International Conference on Distributed Framework and Applications (DFmA)*, 2008.

[13] B. Thuraisingham, L. Khan, M. M. Masud, and K. W. Hamlen, Data mining for security applications, in *Proc. IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC)*, 2008, pp. 585–589.

[14] F. B. Schneider, G. Morrisett, and R. Harper, A language-based approach to security, in *Proc. Informatics: 10 Years Back, 10 Years Ahead*, 2000, pp. 86–101.

[15] B. Alpern and F. B. Schneider, Recognizing safety and liveness, *Distributed Computing* **2**(3) (1987) 117–126.

[16] K. W. Hamlen and M. Jones, Aspect-oriented in-lined reference monitors, in *Proc. ACM Workshop on Programming Languages and Analysis for Security (PLAS)*, 2008, pp. 11–20.

[17] K. W. Hamlen, G. Morrisett, and F. B. Schneider, Certified in-lined reference monitoring on .NET, in *Proc. ACM Workshop on Programming Languages and Analysis for Security (PLAS)*, 2006, pp. 7–16.

[18] M. Jones and K. W. Hamlen, Enforcing IRM security policies: Two case studies, in *Proc. IEEE Intelligence and Security Informatics Conference (ISI)*, 2009, pp. 214–216.

[19] M. Sridhar and K. W. Hamlen, Model-checking in-lined reference monitors, in *Proc. 11th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI)*, 2010, pp. 312–327.

[20] K. W. Hamlen, V. Mohan, and R. Wartell, Reining in Windows API abuses with in-lined reference monitors, Tech. Rep. UTDCS-18-10, Computer Science Department, The University of Texas at Dallas, Richardson, Texas (June 2010).

[21] A. Sabelfeld and A. C. Myers, Language-based information-flow security, *IEEE Journal on Selected Areas in Communications* **21**(1) (2003) 5–19.

[22] A. C. Myers, Practical mostly-static information flow control, in *Proc. 26th ACM Symposium on Principles of Programming Languages (POPL)*, 1999, pp. 228–241.

[23] P. C. Kocher, J. Jaffe, and B. Jun, Differential power analysis, in *Proc. 19th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO)*, 1999, pp. 388–397.