COPYRIGHT NOTICE

Towards an Implementation of Information Flow Security using Semantic Web Technologies

Oana Ureche, Robert Layton and Paul Watters
Internet Commerce Security Laboratory
University of Ballarat
Ballarat, Australia
{o.ureche, r.layton, p.watters}@icsl.com.au

***Abstract*—Controlling the flow of sensitive data has been widely acknowledged as a critical aspect for securing web information systems. A common limitation of previous approaches for the implementation of the information flow control is their proposal of new scripting languages. This makes them infeasible to be applied to existing systems written in traditional programming languages as these systems need to be redeveloped in the proposed scripting language. This paper proposes a methodology that offers a common interlinqua through the use of Semantic Web technologies for securing web information systems independently of their programming language.**

Web vulnerabilities; Computer Security; Information flow; Semantic Web

I. INTRODUCTION

Cyber-crime is a major interdisciplinary concern thriving on the opportunities that information and communication technologies (ICT) offer [1]. Activities which fall under the cyber-crime category expand from the use of malware to gain sensitive information [2] to the use of technology in order to plan terrorist attacks [1]. In this research the authors focus on testing web applications for web vulnerabilities that can lead to cyber-attacks.

Web applications constitute a major target for computer "hackers". High profile companies, such as: NASA [3], Sony [4] and Citigroup Inc. [5], have suffered cyber-attacks that lead to these hackers gaining access to unauthorized data and other information (such as bank account and credit card details, contact information, email addresses) that could lead to identity theft [6]. According to security company McAfee, the last 5 years have seen the biggest series of cyber-attacks in history, including the infiltration of 72 world organizations as well as the United Nations [7].

These series of cyber-attacks are due to implementation flaws in the web applications' underlying programming code. In practice, developers of applications that contain sensitive/hidden data will surround the implementation code with security features to protect the sensitive data. Unfortunately, it is impossible to guarantee that a piece of complex software does not contain some flaws [8]. Developers make mistakes, which results in implementation bugs that can be exploited in order to gain access to unauthorized information. Local implementation errors, interprocedural errors or even malicious programmers trying to compromise the security of the network can cause security breaches [9].

A significant amount of research has been carried out in the area of finding vulnerabilities in web application systems that could expose confidential data by employing methods of information flow control [10]. Most of the proposed research methods use the decentralized label model (DLM) and apply it using a generalization of security classes (labels), security policies and rules describing how data can be consumed based on the security policy that describes the data. A common limitation of these systems is their proposal of new scripting languages. This makes it infeasible for the proposed methods to be applied to existing web applications or systems, as applications will need to be rewritten in the proposed programming language [11].

This research aims to develop a method modeling vulnerabilities in applications through information flow control. This approach will provide not only comparable reliability in terms of the number of vulnerabilities found, but also a method that does not require existing systems to be redeveloped. To evaluate the proposed method, it is compared against previous approaches; specifically the method will be evaluated in terms of number and type of vulnerabilities found. As previous approaches in the area of finding vulnerabilities in computer applications through information flow control are applied to web applications, the proposed method will target web applications and traditional scripting languages: PHP, Perl, and Python.

The key aspect that differentiates the proposed method to previous approaches is that rather than proposing a new language, the proposed technique will be applicable to existing programs. This approach will be based on open standards and W3C recommendations, specifically Semantic Web technologies. Application code will be exported to the Resource Description Framework (RDF) format. A semantic reasoner will detect vulnerabilities by finding inconsistencies in the model, in conjunction with an ontology about the domain of information flow control of sensitive data and rules about how data can be consumed based on their labeling. The proposed method will statically check information flow within web information systems that contain sensitive data.

As a definition, a web information system consists of one or more web applications.

II. BACKGROUND

In the introduction section it was shown that a viable approach to finding vulnerabilities in web information systems is through the information flow control. Controlling the flow of sensitive data has been acknowledged as a critical aspect of securing applications, even at industry level [12]. This section reviews and explains language-based information flow control systems and their limitations in terms of applicability.

A. Language-based Information Flow Control

1) Ad-hoc mechanisms.

Taint mode checking is a feature of some scripting languages, such as Perl and Ruby, used to detect vulnerabilities in applications [13]. Such mechanism treats all input data coming from outside the application and used in potentially dangerous operations, such as writing to a file or make system calls, as tainted values. Safe operations, such as `print` [14] are ignored by taint mode. Malicious users of web applications can exploit security weaknesses by using HTML web forms to input data that will cause the application to behave in an unexpected way. This can lead to access of confidential data. Best practice is to use bind variables by employing prepare and execute statements [15].

In Ruby, there are five safety levels, given by the value of `$SAFE` variable, which provides much finer control over the security checks of the code [16]. The safety levels range from level 0, where Ruby does not perform any checks, to level 4, where non-tainted objects cannot be modified.

Not all scripting languages have a taint mode; Python and PHP are two examples. One advantage of taint mode is that for some languages, its implementation is straightforward. Perl uses an easy algorithm to run the security checks, but is known to contain bugs [11]. The disadvantage of current taint mode implementations is that they provide only an ad-hoc mechanism for detecting vulnerabilities and are language dependent. For example, Perl's taint mode cannot be applied to PHP code.

2) Other scripting languages.

Yip, Wang, Zeldovich and Kaashoek [17] introduced a new language runtime called RESIN that allows, through data annotations at code level, to find and prevent security vulnerabilities in web information systems. The proposed programming language introduces three mechanisms: policy objects, data tracking and filter objects. In order to control data flow, RESIN uses the policy objects to annotate and track sensitive data and then invokes the filter objects when data is written outside the program, such as when writing data to a file or the network. The authors explain how RESIN can prevent a series of vulnerabilities such as SQL injection, server-side script injection, password disclosure.

Although RESIN is effective at finding most security vulnerabilities, the programmers are faced with extra overhead, as they are required to write extra code in order to implement the aforementioned data flow assertions. In terms of application overhead, RESIN adds a 33% CPU overhead when generating a page for a system like HotCRP. However, in their publication the authors did not mention the application overhead for web applications other than HotCRP. Furthermore, RESIN has a potential concern since the data flow assertion can be duplicated by data flow checks and security checks that already existed in an application.

Myers [18] describes JFlow, a new language that extends the Java programming language developed to protect the confidentiality and integrity of sensitive data mostly using statically-checked data flow assertions. JFlow's goal is to prevent sensitive information from being leaked through computation and it achieves this using decentralized label modeling (DLM). As opposed to other approaches for protecting confidential information that use dynamic security classes [19], JFlow uses static checking, allowing a detailed tracking of security classes without incurring run-time overhead. The author provides programmers with a JFlow compiler that will statically check programs written in JFlow. Furthermore, JFlow supports the development of secure applets and servers that handle sensitive data. Although JFlow addressed some of the limitations of previous proposed approaches, (e.g. mutable objects, subclassing, exceptions) programmers are required to write applications in the JFlow language. Therefore, this approach cannot be applied to traditional scripting languages such as PHP, Python or Perl, used in the development of web information systems.

Li [20] proposes a new scripting language and a variable type checker in order to provide security and integrity of data in standard web applications systems that use query languages, such as SQL, to access the underlying database. The language interprets conventional security levels associated with data (e.g. public, secret, untainted, and tainted) according to downgrading policies, where every security level has a policy that specifies what computation is needed to downgrade data to that security level. The scripting language is similar to PHP and it uses the `<?ssp_header` and `!ssp_header>` delimiters for the programming code. The major difference be-tween PHP and the proposed language is that the queries to the web application's database are strongly typed, therefore checking the code for confidentiality and integrity purposes is easier for programmers and the security checker used by the author can automatically determine if the code satisfies the security policies. However, similar to other approaches [18] [21] the author proposes a new scripting language thus, making it difficult for programmers that use standard web application development languages (such as PHP, Perl, Python) to employ this method.

B. Enforcing Security and Privacy Policies Using Semantic Web Reasoning

In this section, an example of a framework [22] that uses Semantic Web reasoning for enforcing context-sensitive security and privacy policies is given. Although this framework is applied in the context of a particular type of Policy Enforcing Agent (PEA), specifically an Information Disclosure Agent (IDA), it is important to note the

methodology used, as conceptually this could be applied in the context of information flow control.

The authors apply their proposed framework to an application scenario where there is a need for enforcing confidentiality and integrity policies in decentralized computing environments [22]. In such a scenario, a person who owns information sources (sensor, user, application etc.) invokes an IDA in order to disclose information ac-cording to pre-defined rules. An IDA receives a request for information. The IDA satisfies that request with information that the privacy and security policies allow to disclose. For example, an employee of a company wants to know the location of an-other employee. The employee uses the framework to send a request for the location information. The IDA of the other employee receives the incoming request, but answers the request based on the privacy and security policies that are associated with that information. The IDA could respond only with the building location and not the actual level in the building where the employee is located.

The authors use RDF, rules and ontologies to describe the data, the policies and the domain knowledge, respectively. ROWL, an extension of OWL, is the specification used for expressing the concepts' security and privacy policies [23].

In this section, it was shown that reasoning with rules and ontologies in order to enforce security and privacy policies was previously proposed in literature. As part of the evaluation of the proposed Semantic Web framework, the authors implemented an example that used 22 rules and 178 facts and that invoked a semantic service directory containing 50 services. Although this framework works in different contexts than the information flow control, it was shown that it is practical to enforce security policies using Semantic Web technologies by reasoning with rules and ontologies.

III. Overall Methodology and Architecture

The limitation of rewriting existing code to account for information flow constraints was previously mentioned by Zdancewic [11]. He recommends a mostly static approach, such that software is analyzed in order to check if policies applied to data are obeyed. In light of this, this paper proposes a methodology that addresses the limitation of previous approaches for securing web applications. Specifically, the proposed methodology leverages Semantic Web technologies in the field of information flow control to develop a tool deployable for existing web information systems, without the need for their code to be rewritten in a proposed programming language.

Fig. 1 shows the overall architecture of the proposed methodology. The inputs consist of web information systems written in different programming languages. The programming code together with the respective language grammar is fed into an AST builder tool, such as ANTLR, and converted to RDF.

Information flow control rules previously described in literature are expressed in RDF. There is a vast amount of work regarding the definition of rules for security policy enforcement [10]. This paper's methodology reuses existing label-checking rule definitions. The inference engine is implemented using the Jena framework inference machinery. The Jena framework constitutes the programming toolkit for the methodology described in this paper. Jena was chosen as it is the most widespread Semantic Web framework [24]. While there are other frameworks, it falls out of the scope of this research to analyze which framework is the most productive or usable.

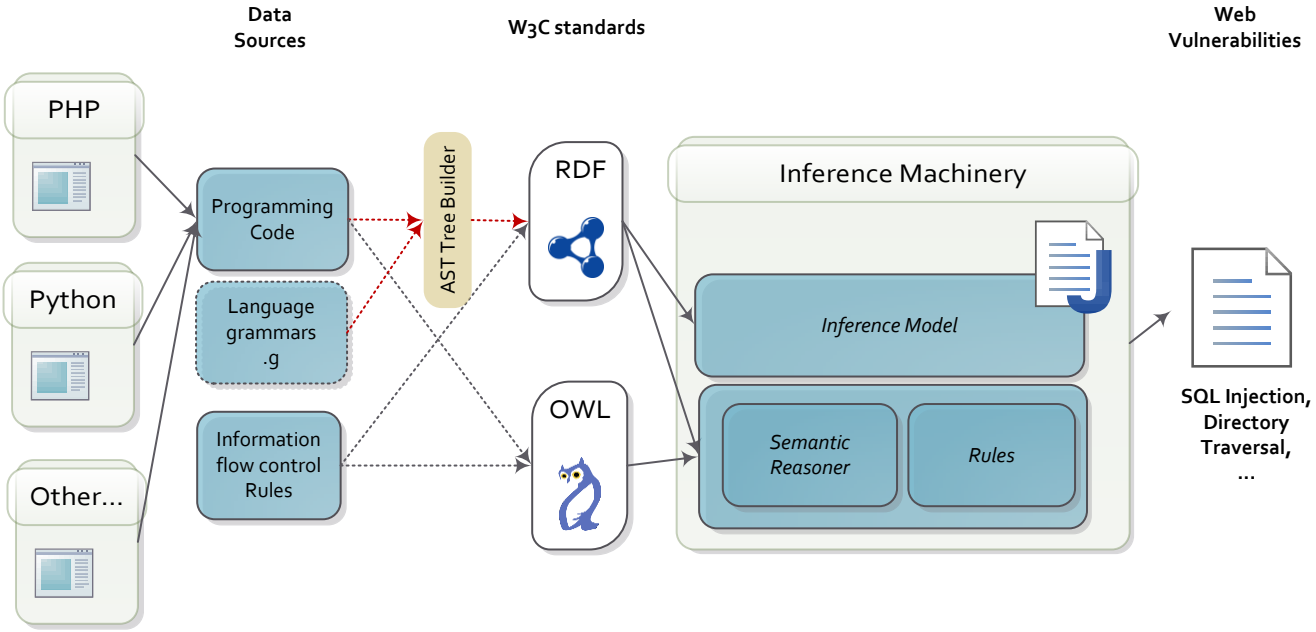

Figure 1. Overall architecture

Using Jena's `ModelFactory` class a new inference model is created by associating a data set with some reasoner. The new model can be queried for statements. When the model is queried for data, statements that originally existed in the data set will be returned together with statements that were not in the original dataset. The new statements are inferred by the semantic reasoner using the defined rules or by other inference mechanisms. Reasoning is achieved using rules and ontologies in the context of information flow control over the programming code translated to RDF. The inferred statements will represent web vulnerabilities found in the web information systems given as input.

IV. EXAMPLE SCENARIO

As aforementioned in section III, the information flow control rules are translated from literature using Semantic Web technologies. The base for the creation of the ontology, the rules and the reasoning is JFlow [18]. First, JFlow is the first practical approach to a language-based information flow control system [18]; and second, this proposed language represented the base for several other proposed approaches: [25], [26] and many more.

JFlow uses a label model in which the label of some data consists of a set of policies and according to these defined policies the movement of the data is restricted.

In this section a simple PHP example that leaks information will be given. The code will be converted to RDF using the JFlow label model. The rule that assures no leakage of information will be expressed using the Jena framework. The following code shows the PHP code. The conversion to RDF follows.

```
<?php
      $b = True; //secret value
      // set of instructions that may
      change the value of b
      $x = 0; //public value
      if ($b) {
            $x = 1;
      }
?>
```

Code fragment 1. PHP code that leaks information.

In this example, `$b` is a secret value and `$x` is a public value. The assignment `$x = 1` leaks information, because if the assignment takes place the value of `$x` changes and thus, the conclusion that `$b` is `True` can be drawn, which gives out publicly the value of `$b`. These public and private variables can be expressed using RDF. These variables will have a label with a policy attached to it that will express their state: public or private.

The information leak rule states that:

"Any assignment where the pc counter has a private policy and it is assigned to a public variable is an information leak."

The pc counter concept is explained in [18]. JFlow associates the pc counter with every assignment and expression.

```
<rdf:RDF
      ...
      <rdf:Description rdf:about="http://flowcontrol.org/1#assignment">
            <fc:id>1</fc:id>
            <fc:variable_value>True</fc:variable_value>
            <fc:variable_name>b</fc:variable_name>
            <fc:hasPolicy>private</fc:hasPolicy>
            <rdf:type rdf:resource="http://flowcontrol.org/Assignment"/>
      </rdf:Description>
      <rdf:Description rdf:about="http://flowcontrol.org/2#assignment">
            <fc:id>2</fc:id>
            <fc:variable_value>0</fc:variable_value>
            <fc:variable_name>x</fc:variable_name>
            <fc:hasPolicy>public</fc:hasPolicy>
            <rdf:type rdf:resource="http://flowcontrol.org/Assignment"/>
      </rdf:Description>
      <rdf:Description rdf:about="http://flowcontrol.org/3#assignment">
            <fc:id>3</fc:id>
            <fc:variable_value>1</fc:variable_value>
            <fc:variable_name>x</fc:variable_name>
            <rdf:type rdf:resource="http://flowcontrol.org/Assignment"/>
      </rdf:Description>
      ...
</rdf:RDF>
```

Code fragment 2. Translation to RDF using the JFlow label model.

The information that can be leaked after the execution of the statement will be assigned to the pc counter. In the PHP code example, at the `$x = 1` assignment the pc counter will have the value `b`. Because `x` is public and `b` is private, the assignment represents an information leak. The rule that states that the assignment is an information leak in this case, is expressed using Jena rule syntax and is listed as follows.

```
//defining namespaces
@prefix fc: http://flowcontrol.org/
[informationLeak: (?d rdf:type
      fc:InformationLeak)
<-
(?d rdf:type fc:Assigment)
(?d fc:pc_counter ?pc)
(?d fc:variable_name ?x)
(?pc fc:hasPolicy "private")
(?x fc:hasPolicy "public")]
```

Code fragment 3. Information leak rule expressed using Jena rules syntax

This example scenario section showed through an information leak example, how an approach using a proposed programming language to annotate data and detect an information leak can be translated using Semantic Web technologies. As mentioned in section III, there is a vast amount of work regarding the definition of rules for security policy enforcement. The proposed approach will apply the same steps to define and apply other rules for security policy enforcement, as it was shown for the PHP code example.

Using the Jena framework, reasoning can be achieved by invoking the Jena inference engine. When the inference model is queried for data, statements that originally existed in the data set will be returned together with statements that were not in the original dataset. The new statements are inferred by the semantic reasoner using the defined rules or by other inference mechanisms. For example, when considering the information flow control code listings examples, one inferred statement could be "Assignment 3 is an information leak" and the representation in RDF follows. In this manner, a list of vulnerable code assignments is returned.

```
@prefix fc: http://flowcontrol.org/
<rdf:Description rdf:about=
      "fc:3#assignment">
      <rdf:type rdf:resource=
      "fc:InformationLeak "/>
</rdf:Description>
```

Code fragment 4. Assignment 3 is an information leak expressed using RDF.

V. COMPARISON

In this section, we provide a qualitative comparison of our proposed technique by comparing it with other popular methods of finding web vulnerabilities. For this purpose, we divide the existing approaches in 4 categories: taint mode, security-typed programming languages, security enforced in the database and our proposed approach.

Our proposed approach offers several desirable features, including the language independent aspect and the machine learning capability. Although security enforced in the database can be considered programming language independent, it is highly infeasible that two or more web information systems written in different programming languages will have the same structure and access for and to the underlying database.

TABLE I. FEATURE COMPARISON BETWEEN SEVERAL DIFFERENT APPROACHES

	Proposed approach	**Taint mode**	**Security-typed programming languages**	**Security enforced in the database**
Language independent	✓	✗	✗	? (with respect to the database)
Machine learning capability	✓	✗	✗	✗
Reusable/ Modular	✓	✓	✓	✗
Effective	N/A	✓	✓	✓
Multiple levels of security	✗ (but possible)	? (with respect to the scripting language)	✗	✗
No complex implementation	✗	✓	✗	✗
No new scripting language	✓	✓	✗	✓

VI. CONCLUSIONS AND FUTURE WORK

Due to the nature of web browsers and the HTTP protocol, web information systems are known to be highly insecure. “Hackers” target these systems in order to expose confidential information such as usernames and passwords leading to major losses in revenue and loss of users trust. While existing approaches that find web vulnerabilities are effective at discovering security pitfalls, they are language dependent. Thus, for different web information systems potentially different methods need to be applied, making existing methods not scalable. This paper describes an original technique that finds web vulnerabilities in web information systems and is independent of the programming language used and capable of machine learning using inference engines.

Future work includes the implementation and the quantitative evaluation of the proposed technique. To evaluate we will use popular web information systems written in several different programming languages (e.g. PHP, Perl, Python and Java). We will use older versions of these systems with a list of security issues discovered in the past. The evaluation report will conclude that the proposed technique is effective at detecting vulnerabilities in web applications once all the known vulnerabilities are found. Yip, Wang, Zeldovich and Kaashoek [17] used this type of evaluation.

ACKNOWLEDGMENT

This research was conducted at the Internet Commerce Security Laboratory and was funded by the State Government of Victoria, IBM, Westpac, the Australian Federal Police and the University of Ballarat. More information can be found at the Internet Commerce Security Laboratory website[1].

REFERENCES

[1] K-K. R. Choo, “The cyber threat landscape: Challenges and future research directions”, Computers & Security, pp. 719-731, 2011.

[2] M. Alazab, S. Venkatraman, and P. Watters, “Zero-day Malware Detection based on Supervised Learning Algorithms of API call Signatures”, Ninth Australasian Data Mining Conference: AusDM 2011, 2011.

[3] Bloomberg - Business & Financial News, Breaking News Headlines, “Bloomberg L.P. Network Security Breaches Plague NASA”, http://www.businessweek.com/magazine/content/08_48/b4110072404167.htm.

[4] The Sydney Morning Herald, “PlayStation hacking scandal: police chief says contact your bank now”, http://www.smh.com.au/digital-life/games/playstation-hacking-scandal-police-chief-says-contact-your-bank-now-20110427-1dvts.html.

[5] Business Spectator Pty Ltd., “Citigroup cyber security breached”, http://www.businessspectator.com.au/bs.nsf/Article/UPDATE-3-Citi-says-hackers-access-bank-card-data-HNAQP.

[6] A. Stabek, P. Watters, R. Layton, "The Seven Scam Types: Mapping the Terrain of Cybercrime," Cybercrime and Trustworthy Computing, Workshop, pp. 41-51, 2010 Second Cybercrime and Trustworthy Computing Workshop, 2010

[7] Guardian News and Media Limited, “Biggest series of cyber-attacks in history uncovered”, http://www.guardian.co.uk/technology/2011/aug/03/biggest-series-cyber-attacks-uncovered

[8] M. Gasser, “Building a secure computer system”, New York : Van Nostrand Reinhold Co., 1988.

[9] McGraw, G., & Potter, B. (2004). Software Security Testing. IEEE Security and Privacy, Vol. 2 , 81-85.

[10] A. Sabelfeld and A. Myers, “Language-based information-flow security”, 2003, Selected Areas in Communications, IEEE Journal on, pp. 5-19.

[11] S. Zdancewic, “Challenges for information-flow security”, In Proc. Programming Language Interference and Dependence (PLID), 2004.

[12] Sophos, Inc. e-Guide, “Evolving IT security threats. Inside Web-based, social engineering attacks”, http://searchsecurity.bitpipe.com/detail/RES/1308928423_477.html.

[13] W. Masri, A. Podgurski and D. Leon, “Detecting and Debugging Insecure Information Flows”, 2004, Software Reliability Engineering, International Symposium on, pp. 198-209.

[14] A. Hurst, “Analysis of Perl’s Taint Mode”, http://hurstdog.org/papers/hurst04taint.pdf.

[15] OWASP, “SQL Injection Prevention Cheat Sheet”, https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet.

[16] D. Thomas and A. Hunt, “Programming Ruby: A Pragmatic Programmer’s Guide”, s.l. : Addison-Wesley, 2000.

[17] A. Yip, X. Wang, N. Zeldovich, M. Kaashoek, “Improving application security with data flow assertions”, Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles. New York : ACM, 2009, pp. 291-304.

[18] A. Myers, “JFlow: Practical Mostly-Static Information Flow Control”, 1999. In Proc. 26th ACM Symp. on Principles of Programming Languages. pp. 228-241.

[19] C. McCollum, J. Messing, and L. Notargiacomo, “Beyond the pale of MAC and DAC-defining new forms of access control”, Oakland, CA , USA : s.n., 1990. Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on. pp. 190-200.

[20] P. Li, “Practical information-flow control in web-based information systems”, In Proceedings of 18th IEEE Computer Security Foundations Workshop. IEEE Computer (pp. 2-15), s.l. : Society Press, 2005.

[21] V. Simonet and I. Rocquencourt, “Flow Caml in a Nutshell”, Proceedings of the first APPSEM-II workshop, pp. 152-165, 2003.

[22] J. Rao and N. Sadeh, “A Semantic Web Framework for Interleaving Policy Reasoning and External Service Discovery”, 2005. In Proceedings of RuleML, pp. 56-70.

[23] F. Gandon, and N. Sadeh, “Semantic web technologies to reconcile privacy and context awareness”, 2004, Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 1, No. 3. pp. 241-260.

[24] F. Lindorfer, “Semantic Web Frameworks”, Basel : Basel University, 2010.

[25] S. Zdancewic, “Practical Information-flow Control in Web-Based Information Systems”, 18th IEEE Computer Security Foundations Workshop CSFW05 (pp. 2-15), s.l., IEEE, 2005.

[26] R. Indrajit, D. Porter, M. Bond, K. McKinley, and E. Witchel, “Laminar: Practical Fine-Grained Decentralized Information Flow Control”, Proceedings of the 2009 ACM SIGPLAN conference on Programming language design and implementation, pp. 63-74, s.l. : ACM, 2009.

[1] http://www.icsl.com.au