

## SOA Web Security and Applications

**Raymond Wu**, Department of Research and Development, NST, Inc, Japan  
**Masayuki Hisada**, Department of Research and Development, NST, Inc, Japan

### Abstract

The conventional vulnerability detection fails to extend its generic form to an abstract level in coping with particular type of string validation. Consequently the security bypasses key issues such as Java scripting and SQL injection. It causes tremendous business loss and customers risk due to taint distribution and illegal data manipulation. This paper introduces semantic analysis by using metadata codes, as well as a hierarchical parser in token-based algorithmic check. Our research in SOA web security can help industry to minimize business impact, to achieve higher accuracy in vulnerability detection, and to commit fast responsiveness.

## 1 INTRODUCTION

To deal with taint prediction, we proposed token based metadata to validate semantic notation. We introduce SOA service bus and token based framing technique as supplementary solutions to strengthen token based architectural foundation. This enables the parser analyzer to isolate semantics from the frame in performing input string validation. In service industry, as componentization and our token based strategy encapsulate robust validation and tracking, point of failure can be precisely identified at transaction level, and business impact can be minimized. We aim to investigate the key factors of web security to encompass the research work. Among these key factors, the identification, validation and tracking (IVT) are all our research milestones. In IVT process, our approach in identification was sound and the deployment has been smooth. This enables node identification, run-time metadata capturing, knowledge-based repository, and client/server messaging. The validation process deals with token decomposition, parser code generation and variables validation. The process applies automaton in state transfer, so the string can be decomposed and validated against a rule-based policy. Finally, the tracking process explores the results from identification and validation from a comprehensive macro view standpoint. It involved in metadata analysis, suspects prediction, and knowledge-based repository construction. It predicts those suspect event and data transmission over flow graph, and performs metadata messaging for tracking and visualization.

As it was indicated in our previous research, in order to achieve a full coverage of web security, we apply the common abstract syntax to produce nodes structure as basic element [1]. This enables us to identify an arbitrary physical element by using a symbolic notation. To shed more light on the architecture view in flow analysis, the flow ontology is classified into macro and micro architecture. Macro view stands for the system level topology such as control flow analysis or data tracking over node configuration, while micro view deals with the interoperability beneath the node level, such as string validation automata [2]. The contribution of our vision has achieved many successful deployments based on the unique visions of enterprise coherence between backbone and branches.

## 2 SOA TREND AND SECURITY APPROACHES

Web security is not an independent area within enterprise architecture; instead it has played significant role in enterprise transformation and has been subset of IT trend. In order to strengthen the security architecture, an understanding of IT trend and the architectural milestones is necessary. In reviewing the IT trend in recent 50 years, the key vendors endeavor to support industry to minimize integration effort and business risk, and to consolidate heterogeneity into the homogeneity, thus the standard, open-interfacing and consolidation have become a common practice in the industry. Figure 1 reflects the trend and evolution milestones.

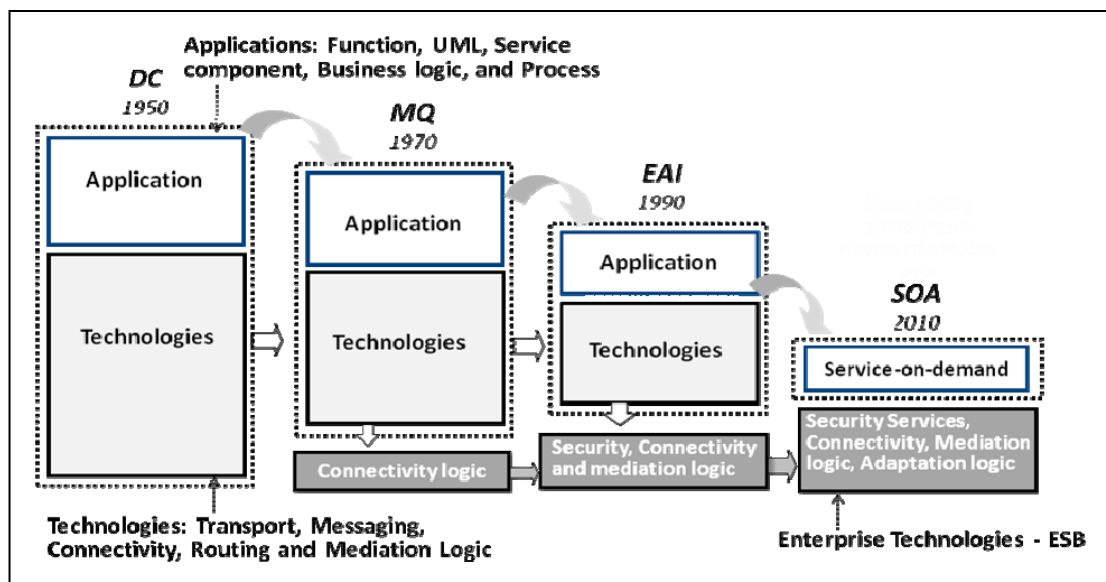
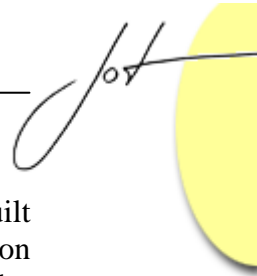


Figure 1. Trend of security and service industry

The Application follows top-down approach to streamline the services and process implementation, so the service providers can play mediator role between service provider and consumers. In contrary to application, technologies take bottom-up approach from the convergence of ad hoc technologies. Key vendors such as IBM, SAP, Oracle, HP,



---

Microsoft and Tibco have launched service oriented architecture (SOA) and built Enterprise Service Bus (ESB) middleware to support SOA technologies. So the common services such as adaptability, security, protocol, format, logic and routing can be under ESB's centralized control [3].

The concerns have been raised, as SOA achieves enterprise strategy, however ESB security doesn't commit full coverage of attacking patterns. The ESB transport has been built on top of trusted relationship between invoker and service through Virtual Private Network (VPN) and SSL, thus security is limited to a role based "systematic protection" instead of "transactional prediction". The current ESB security protects resources from attackers for that system level accessibility; so far it has not been enabled to detect semantic bypassing cases.

Our research about three layer's interoperability attempted to complement weakness of ESB security. Our security architecture takes advantage of ESB features in adaptor, transport and routing services as technical foundation, and proposes add-on services such as abstract syntax, trigger embedding, metadata messaging, parser automaton, and light-weight parser code to enhance semantic analysis. Our add-on security services compensate weakness of ESB in collaborating functional elements and support static and dynamic analysis.

Our approaches are first, to extract abstract syntax tree (AST) for building up generic format, and to apply event trigger plug-in for flagging and logging in creating node identification. Generic format means that the metadata about information, algorithms and visualization are all in same format which is language independent. This allows the universal adaptor, once finish language recognition, to initiate the parser, to analyze the syntax, and to generate a generic syntax for flagging purpose. The objective of flagging and plug-in is trying to embed the message into each program block, so we can create the linkage between nodes, the run-time path, and the necessary message produced at run-time logging. The messaging features move identification strategy forward in mediating macro and micro architecture view. It creates linkage between metadata in symbolic form and descriptive node information. To this ends, we embed the event triggers as plug-in of the codes to capture necessary information for analysis. The messaging features mediate macro and micro architecture view and create linkage between symbolic metadata and descriptive node information. By using this embedding mechanism, the event trigger generates metadata message at run-time when the node is executed.

Our metadata strategy is to apply three forms of metadata approach. It first deals with language heterogeneity and different level of abstract to build up unique semantics mediation. The three forms of metadata consist of Messaging Based embedded trigger Metadata (MBM), Token Based parser automata Metadata (TBM), and Knowledge Based traceable & predictable Metadata (KBM). Basically these three types of metadata interoperate closely with each other. MBM deals with key identification process, TBM handles most validation process, and KBM manages major tracking process. As indicated in Figure 2, all of three type's metadata are stored in KBR. MBM can be in the form of symbolic notation as a pointer, structure of systems, or the abstract of a data object. The message, embedded in the statement block as a node, capture the snapshot information

when executed, so the logging messages can be generated at run-time process in HTML form, and represent the traces of an individual execution path [4]. TBM works on the decomposition of tokens, so each token is encoded to a special parser code, it will then be filtered and integrated into string code. The formation of a string code can be representation of semantics. It differentiates multi-dimensional approach from conventional process by encapsulating new attributes of light-weighted metadata [5]. Our “meta on meta” mediates multi-types of metadata and build up a transparent environment for security analysis.

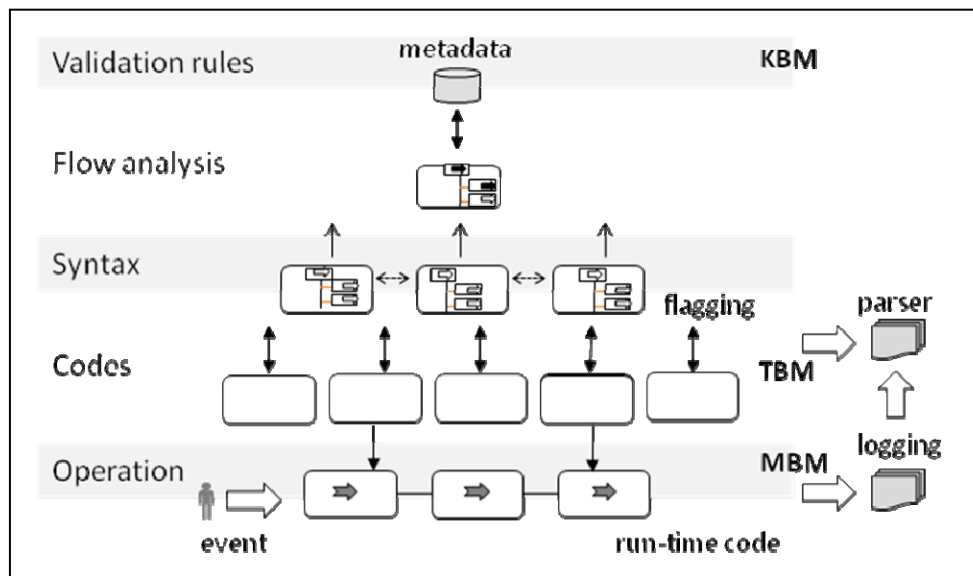
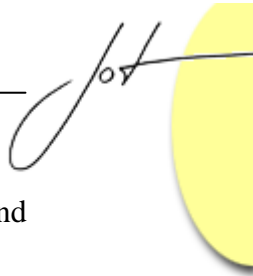


Figure 2. Hierarchical structure of language abstract

### 3. TECHNIQUES AND APPLICATIONS

Framing techniques and Meta code have been applied in the research to support our three forms metadata approaches. Among the attacking patterns, SQL vulnerability has been recognized as one of significant web security issues hard to detect and to sanitize. Conventional methods have been weak in identification and taint control at source site. The SQL attackers take advantage of the hidden variables in bypassing string validation furthermore, the blind point of syntax check leaves backdoors for tautology, sensitive characters, and SQL injection [6] has been used as vehicles in SQL attacking.

Framing and filtering approach was developed as foundation of semantic analysis to take Parser Automaton (PA) and messaging in generating Meta code. PA can solve most syntax level vulnerability by comparing the PA code before and after malicious data is entered. As each token is associated with a PA code (i.e. “00” for basic frame, “21” for “=”, “11” for “and”), and each PA code apply a hierarchical level code (starting from 0) to represent its vertical level, the integration of PA code and level code form a hierarchical PA (HIPA) string, the example is as followed; SQL sample: select name from customer



---

where city\_code = (select code from city where city\_name = "<inp1>" ) and group\_code="<inp2>";

HIPAA code: "0.00-0.21-1.00-1.21-2.92-0.11-0.21-1.92-0.31"

To determine the Meta code for comparing, the first step is based on pure SQL frame (SF), the second step works on the tokens of SQL variables (SV) filtered from frame, and the third step takes the whole token based metadata of SQL structure (SS) into account. Experiments led us the results that the validation process may bypass any step from three, but the hidden malicious data has difficulty in bypassing all three steps of SF, SV and SS validation. This semantic validation method differentiates its approach from system level or pattern matching approach as the later one take only SQL input string (SI) into account, even though the SQL code (SC) itself stay fixed, the SI validation bypasses imitation of truncated SC and makes SS unchanged. The objective of framing process is to separate "engineering" parts from "semantics" parts, so it will not be tainted by hidden variables. Our main task is to work on SC from source code, to identify tokens which constructing the frame, and determine SF by a key words search framing process.

In industry applications, Fraud Management in Service Provisioning has become one of the focuses in industry research. It is very common that in the service industry, many security functions were implemented within a large program, or embedded in a platform such as firewall, or coarse-grained services inside the portal. Apart from validation capability, concerns have been raised from current security features as they may cause tremendous revenue loss and loyalty impact due to a mere suspicious transaction. Among industry sectors, the financial industry is one of the most highly complex among the various sectors and along with the complexity comes high risk caused by security issues. Very often those "suspicious transactions" may enforce the whole program pending or the user account be suspended until the transaction is rolled back and status is clear To eliminate the risk, our assumption is: The higher occupancy of loosely-coupled common services within the enterprise, the less suffering will be caused as we can easily identify the source with effective change. It helps a quick identification of the linkage between source and sink; it also supports scoping of issue area for plug-in services to fix the problem effectively.

Our investigation indicates the weak prediction, late responsiveness; fatal binding and lack of precise scoping in a mixed-up implementation are all common issues of web security. In the operational terms, these are related to the weakness in prediction, detection and issue-handling. The common access point for event tracking facilitates unique tracking mechanism for risk prediction [7]. Once the transaction matches a specific pattern stored in KBP, it will be marked as "suspicious" and the event tracker invokes security services to validate the transaction. Since prediction has no opportunity to check details of the transaction and the data input, it is only limited to an empirical statistics based on user behavior and the process routing. The intelligence of the statistics can be encoded to a light-weighted Meta code similar to HIPA, so the Meta code generated at portal adds-on service can be compared to Meta code stored in KBR.

The prediction features provide a precise detection when the security service is invoked to perform semantic validation. The risk level indicator assesses the user profile,

the incoming event and process routing to determine the risk level and validation strategy. Once the security services is invoked to perform validation, the PA will work on the token based decomposition, and Meta code generation. Figure 3 further illustrates the malicious data detection by either HIPA code or visualized graph, it can be easily detected for those changed graph once attackers inject malicious data, and try to manipulate original code by either syntax change or semantic violation. The complementary solution of HIPA enhances the vertical structure by comparing the level of the HIPA code before and after malicious data is entered. This is the reason we introduced HIPA to add semantic dimension on top of single dimension that PA formed.

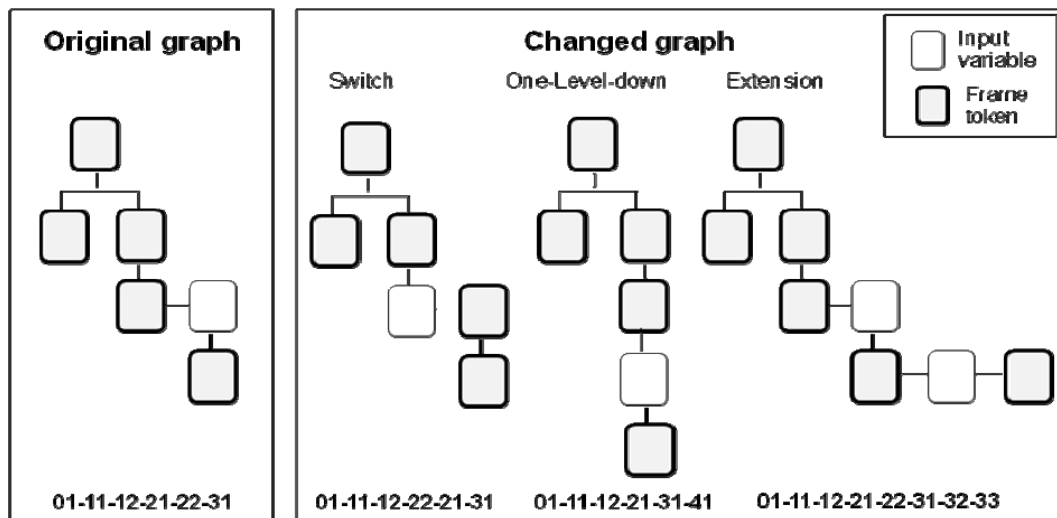
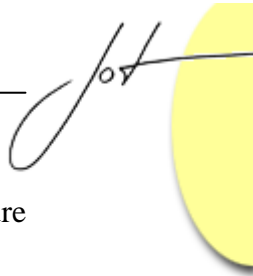


Figure 3. Representation of HIPA code by either digitized code or visualized graph

The framing process supports token decomposition and meta code generation, as the applications can be applied to many key languages. In the conventional credit card service, the inter-banking maintains a common “fraud detection policy” service at system level instead of account level. The framing algorithms are under investigation for how to maintain the integrity of the solid parts of language statements. The violation of frame is detectable by the meta codes, it helps the isolation of security issue from whole system at transaction level. The precise identification of issue works on the event effectively and doesn't cause business impact. The proposed framing process can also cope with generic cases in Cross site scripting (XSS) and SQL Injection (SQLI). We like to take further example of XSS for how the three-layer PVS architecture deals with a multi-sites attacking pattern. Firstly, we studied the mechanism of interactivities. Following SQL injection illustrates the malicious data of SQL injection [8]. XSS is the other common way attackers take advantage of “fragility” of web applications for sensitive data injection. The malicious codes, caused by sensitive data, are generated and embedded into web browser waiting for execution. Basically XSS attackers apply either persistent or ad-hoc models to automatically generate “tainted codes”. The vulnerability can stay in the web browser and wait for victims to execute; it can be embedded into distributed messages and trap victims to click. So once the confidentiality such as cookies or





password intercepted by attacker, it can be abused as means of illegal transaction. Figure 4 illustrates the multi-sites attacking pattern between client, front-end and back-end [9].

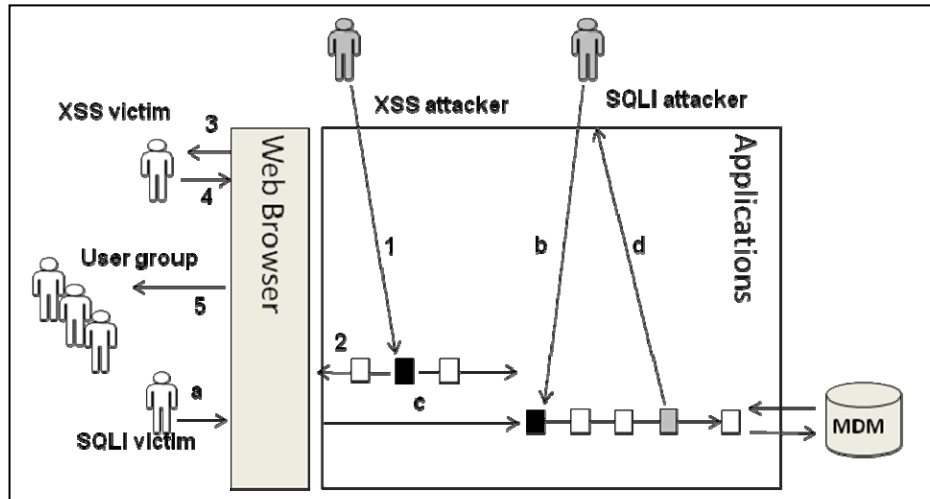


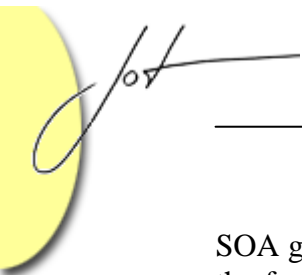
Figure 4. Multi-sites attacking scenarios

To cope with the common issues handling about fraud management, “fine-grained and light-weighted” security services has been targeted as strategy for resolving the SQL injecting issues. It consists of SQL parser decomposition and HIPA hierarchical metadata. It was able to analyze abstract syntax in vertical direction as complementary solution. The conventional syntax check leaves input variable to use tautology, sensitive characters, and injection as vehicles in SQL attacking. Our approach encompasses SQL vulnerability analysis in decomposition and validation. Decomposition first parses source codes into token base semantic notation. This implies those “suspicious transactions” may enforce the whole program pending or the user account be suspended until the transaction is rolled back and status is clear. Security services about SQL attacking patterns are introduced in representing semantic security services.

### 3 CONCLUSIONS

We apply metadata messaging and embed trigger to support identification process. It moves the abstract syntax structure and run-time information into the coherence of token based validation. Our metadata strategy and framing techniques are built on top of the ESB as supplementary fine-grained services of SOA, it distinguishes our approaches from conventional service industry as it provide transaction level detection. The semantic validation, architectural coherence, framing and HIPA techniques are all unique characteristics in positioning our approaches as pilot in SOA web security.

The interoperability of three type’s metadata (MBM, TBM and KBM) provisions our approaches as mediation tools. The token decomposition and separation of frame-variables, enables robust validation by comparing their Meta codes. KBR complies with



SOA governance; it stores algorithms for tracking, and rules for validation. It also applies the feedback from validation to tune the algorithm which is used in prediction, detection and compensation. Our security approaches and development were being realized through industry applications which can be supplementary solutions of ESB. It supports SOA security such as security as service, service on demand, light-weighted Meta code, event-driven invoker, enterprise portal, ESB and componentization of fine-grained add-on services. This paper further introduces semantic analysis for SQL validation by using parser and HIPA, so the input strings, filtered from framing process, forms a hierarchical metadata, to be validated against a token-based algorithmic check. The approach achieves higher accuracy in malicious data detection, and fast responsiveness.

## ACKNOWLEDGEMENT

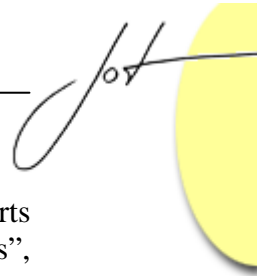
The present research was supported through a program for the promotion of Private-Sector Key Technology Research by the National Institute of Information and Communications Technology (NICT) of Japan, entitled “Research and development concerning the web application security with combination of static and dynamic analysis”.

We also thanks to Professor Atsushi Kara from Nara National College of Technology and Professor Takafumi Hayashi from University of Aizu, for their advisory and information gathering.

## REFERENCES

- [1] TCS, IBM and EDS: Abstract Syntax Tree Metamodel (ASTM), OMG Document (2007)
- [2] Wu, R., “Service design and automata theory”, International Conference on Enterprise Information System and Web Technologies (EISSWT-07) 2007
- [3] Hinton, H., Hondo, M., and Hutchison, B., “Security Patterns within a Service-Oriented Architecture” IBM SOA, <http://www.ibm.com/websphere/developer/services/> Nov. 2005
- [4] Wu, R., Hisada, H., and Ranaweera, R., 2009, “Static Analysis of Web Security in generic syntax format”, July 2009, The 2009 International Conference on Internet Computing, July 13-16 Las Vegas, USA
- [5] Pietraszek1, T., and Berghe, C.: Defending against Injection Attacks through Context-Sensitive String Evaluation, IBM Zurich Research Laboratory and Katholieke Universiteit (2004)
- [6] Halfond, G., and Orso, A., “AMNESIA: Analysis and Monitoring for Neutralizing SQL-Injection Attacks”, ASE 2005, Long Beach, CA, USA, Nov 2005





- 
- [7] Gegick, M., and Williams, L., “Toward the Use of Automated Static Analysis Alerts for Early Identification of Vulnerability- and Attack-prone Components”, Research paper, North Carolina State University
- [8] Liu, A. and Y. Yuan, Y., “A Stavrou, SQLProb: A Proxy-based Architecture towards Preventing SQL Injection Attacks”, SAC March 8-12, 2009, Honolulu, Hawaii, U.S.A.
- [9] Vogt, P. et al. :Cross Site Scripting Prevention with Dynamic Data Tainting and Static Analysis, NDSS (2007)

### About the authors



**Masayuki Hisada** was an assistant professor of the Dept. of Information and Computer Science, Kanazawa Institute of Technology, Japan. He currently serves as director of Dept. of R&D, NST Inc. He received his B.S, M.S and D.S degree in computer science at the University of Aizu, Japan. His research interests are web application security and security technologies for cloud computing. He is the member of IEEE computer society. He can be reached at: [hisada@nstlab.com](mailto:hisada@nstlab.com)



**Raymond Wu** is currently working with NST Inc. in Japan as senior researcher. He has worked with IBM, Oracle, Citibank, Vodafone and the US Government as Lead/Enterprise Architect. He is a member of IEEE, his research interests cover enterprise integration, semantics, web security, computing and database. Raymond received his PhD from UTS, Australia and significantly engaged in SOA and security services. He can be reached at: [raymond@nstlab.com](mailto:raymond@nstlab.com)