The Architectural Review of Web Security in Static and Dynamic Analysis

Raymond Wu and Masayuki Hisada
Department of Research and Development, NST, Inc.
Aizuwakamatsu, Fukushima, Japan

Abstract
Our objective in web security is to move black box to white box in enterprise practices. In this paper, we explain how our approaches achieve the goal in terms of static and dynamic analysis. To better explain the framework and roadmap of analysis work, we describe our approaches by using macro and micro views individually. Based on this foundation, we explore dynamic analysis in string validation and node tracking, and introduce micro and macro views to architect comprehensive approaches. Micro view is related to the mechanism inside the node, so the event triggers and string validation are both under its coverage. Macro view is related to the node tracking which is under investigation of pattern benchmarking. Our evaluation reflects that a configurable and well-tuned topology helps architectural collaboration, consequently it achieve a better security governance. This paper further explains the architectural coherence of identification, validation and tracking. It started with node identification with further exploration to the issue identification.

Keywords:
vulnerability, web security, validation, tracking, dominant, static analysis, dynamic analysis, automata

1. Introduction

To envision the disciplines of web security, we proposed the following techniques. They can support the analysis in generic format in achieving the goal; 1) to build up abstract syntax tree (AST) in syntax format to take most key language codes for analysis, 2) to provision event trigger common practices, and apply flagging and logging techniques in node identification, 3) to implement messaging carrier as common format in metadata transmission, 4) to explore event based automaton for input string validation. These techniques rely on a specific technical architecture for realization. From the overview of technical architecture requirements, our security approaches of static and dynamic analysis, is divided into front end and back end processors. Front end processor manages incoming event, and provide adaptive automation, syntax abstraction, and flow analysis services. The back end server handle the computing algorithm, string validation and repository management for extensive analysis.

Manuscript received August 5, 2009
Manuscript revised August 20, 2009

Web security has caused tremendous disaster in the decade due to the ambiguity of issue identification and tracking mechanism. Furthermore, the solutions are not in a collaborative mode from the architecture stand points. In reviewing the fundamental disciplines in web security, we investigated the key factors of web security to encompass the research work. Instead of working on a specific domain, our research oversees the major concerns of web security with an architecture overview. Our research produced key concepts in flow, system, process and roadmap. In order to demonstrate the framework it has been built, this paper first try to itemize these conceptual aspects, with descriptive outlines stated as followed:

Flow - In order to achieve a full coverage of static analysis, we apply the common abstract syntax to produce nodes as basic element [1]. This enables us to identify an arbitrary physical element by using a symbolic notation. We further apply string validation automaton to enhance dynamic analysis in addressing real-time data transmission. To shed more light on the architecture view in flow analysis, the flow ontology is classified into macro and micro view. 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. The contribution of our vision has achieved many successful deployments based on the unique visions of enterprise coherence between backbone and branches.

System - From system architecture perspective, our scope covers back-end server, front-end server, and client site as a full domain. The full coverage enhances individual site to perform its specific function at service contract level. In order to perform a full diagnostics, none of these three should be missing as all of them play critical role in security governance.

Process - Our operational view maps the attacking patterns onto the use-cases and works on the resolutions. Firstly, the process of attacking events is considered for both client and server sites. Client site consists of attackers and users. Attackers may engage in security violation directly or indirectly, while users include those victims who are used as vehicle for that illegal transaction. Server site consists of

front-end server and back-end server. The front-end basically deals with metadata activities for flow analysis and adaptive interfaces, while back-end manages the knowledge based repository and intelligence of run-time validation.

Roadmap - By taking the view into practices, we propose a three-step solution as implementation roadmap, which consists of identification, validation and tracking. Our approaches in identification are sound and the deployment has been smooth. The systems are up and running, and performing their expected functions such as; node identification, run-time metadata capturing, knowledge-based repository, and client/server messaging. The validation process which is the second step, is a typical micro architecture process dealing with input string validation. The process applies automaton in state transfer, so the string can be decomposed into token and being 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 input string and data transmission over flow graph, and performs messaging between clients and servers for those ambiguous events.

The reason we proposed three-step approach as a roadmap was that our research aims to turn black box into gray box initially, and to achieve web security governance as the final result in achieving white box. Followed by the introduction, this paper starts with the concept of three-step approach to highlight our milestone in identification process, it then addresses the policy-base validation automata, and explores the feasibility of tracking pattern in terms of the configurable flow graph topology. The paper also takes the experimental benchmarking and modeling, into the architecture review, it finally ended by conclusion.

2. Identification

Node discovery - Our techniques for the static analysis are based on the processes of abstract syntax generation, node identification, and metadata coordination. Empirical experiments and literature review led us to a conclusion that a common abstract syntax is required for identification process. Based on this assumption, the mediation between logical and physical world can be taken place in working toward a coherence of macro and micro architecture. The foundation of robust interoperability supports flow analysis, data validation and vulnerability tracking. Consequently, the deployment of identification process produced a clear picture of static analysis by using pointers as bridge in mediating physical and logical world. Further more, the design applies recursive refining process, to work effectively on the mediation mechanism of metadata messaging. It dynamically produces metadata for static analysis, and takes run-time messaging, and tracking on data transmission into encapsulated knowledge-based information for further analysis.

The knowledge regarding a node is defined like this; the symbolic notation of a node is represented as a string constructed by a node id and attributes such as program name, starting and ending line, and other run-time generated information. Node is a pointer mapped to a physical block of the code which should be identified before any analysis task can be performed. Since node identification build up the foundation of flow tracking as an atomic element, this task was set as our first step in our research roadmap. The techniques in provisioning node identification are crucial as it gives us opportunities to embed the arbitrary messaging trigger into the node structure. By using this embedding mechanism, the event trigger generates metadata message at run-time when the node is executed. The messaging features mediate macro and micro architecture view and create linkage between symbolic metadata and descriptive node information.

Adhered to the construction of abstract syntax analysis, techniques in flagging, logging, and metadata messaging are fundamentals. The objective of flagging and plug-in is 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. We introduced the flagging and logging approach in the initial step of static analysis, has been recognized as a pilot in security identification solution [2]. This is shown in Figure 1.

Metadata strategy for static analysis - Metadata messaging can be in the form of the aggregated data, 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. The capsulation of static and dynamic metadata forms event based messaging, and plays critical role in our analytical approaches from the uniqueness of its generic format and enterprise coherence. To this ends, we embed the event triggers as plug-in of the codes to capture necessary information for analysis.

The messaging format is generic and languages independent. However, in term of abstract syntax, we are still trying to manager the dependencies at this time, in consolidating each language's abstract syntax into one higher format or group of formats. This imply, initially we need to categorize the syntax format into sub-categories such as, java based object oriented (OO) and un-typed PHP based structure programming as separate AST formats, so finally we can consolidate them into a generic AST format. 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 reverse engineering further take the AST, after flagging process, into source code plug-in process, so the embedded triggers can perform the logging as an event trigger for each incoming event, and generate messaging, and flow path semantics. The node notation is built on top of map of the node transfer (S_map) which consists of two basic elements; the program description (P-map) as a pointer of physical program name, block location and level in a hierarchical structure, and the tracking description (T-map) as a snapshot of thread, sequence, language being used, and the status of violation and suspect classification.

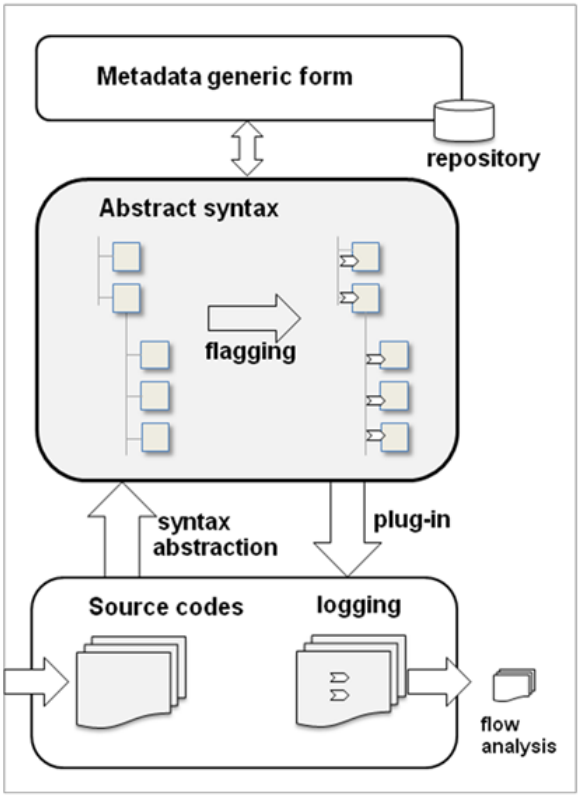


Figure 1 Node Identification process

Table 1 further illustrates the encapsulation of physical, logical, process and semantics factors into a generic metadata format. It is generated once the event trigger is incurred in particular node transfer. Generic format means that the metadata of information, algorithms and visualization are all in same format which is language independent. Given the language dependency in front end analysis, the individual AST apply a generic format through metadata transformation, and commits full transparency in mediation principles.

Table 1. Metadata Messaging generic format

id	Value content	referred value
v1	program id	*refer* static P_map
v2	begin line no.	*refer* static P_map
v3	end line no.	*refer* static P_map
v4	statement code	*refer* static P_map
v5	level	*refer* static P_map
v6	thread id	*refer* real-time T_map
v7	sequence	*refer* real-time T_map
v8	language code	*refer* real-time T_map
v9	pattern code	*refer* real-time T_map
v10	suspect	*refer* real-time T_map

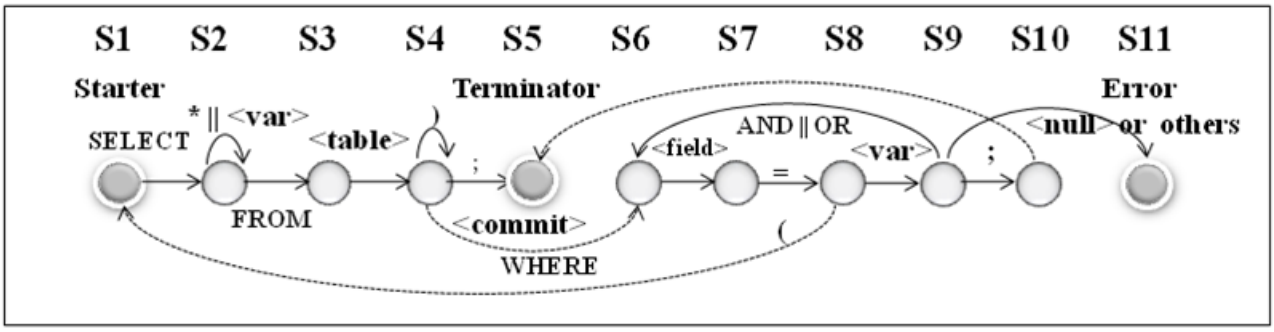


Figure 2. Automata in analyzing SQL based tokens

3. Validation of input data

Reason of input validation - In order to extend the macro view tracking mechanism into micro view detection, we apply automata validation to take the foundation of node based analysis into a micro view. This has been focused on the incoming event tracking and string value validation.
We further introduce a state-transfer automata method, on top of the control flow foundation to deal with the tracking of data transmission. This was completed by some collaboration work between static analysis and dynamic analysis. The approach builds up policy-based validation automata to bridge context-free flow analysis and the semantics of violation identification.
As a complementary of knowledge-based implementation, semantics repository is applied to hold the validation rules, to detect violation and that suspicious path, and to coordinate between macro and micro architecture in maintaining a coherence of security governance. The flow analysis uses node as a pointer, to map the physical world to symbolic notation which apply messaging carrier across ontology of macro view of tracking and micro view of validation [3]. From vulnerability tracking objective, the physical environment require descriptive notation in pushing the AST metadata into a symbolic world as first step, we then identify those "hotspots" onto the flow graph and validate the argument against the flow branch [4].
This gives us opportunity to develop the validation process beneath the control flow, and to realize micro view of data flow tracking through state transfer. In addition, it creates a modulized elements which are reusable which is sharable within enterprise. In order to create an implicit linkage between inter-procedural input and output parameter transfer, a mediator component was proposed to de-couple the binding between sites. Under such circumstances, the data flow transmission between calling, mediator, and called sites form a loosely-coupled architecture for data flow analysis.
State transfer - The coordination of node tracking and state transfer forms a robust coherence of enterprise governance, as we can save system overhead and perform detection for those suspect nodes in extensive investigation [5].
Automata in security component interoperability has similarity in state transfer of Tuning Machine [6]. The state transfer in the string validation applies validation rules which is stored in metadata repository, this is similar to a reading tuning machine in state transfer and output analysis [7] [8]. The alphabets from user input, read in automata, is validated by certain point of recognizer which is leveraged to security event trigger and being executed when the state change is recognized through the knowledge base repository.
Figure 2 illustrates the validation process of following SQL-based vulnerability.

SELECT * FROM mytable WHERE id=? and pwd=? $id = addslashes($_GET['id']); $pwd = addslashes($_GET['pwd']); (Id = "1 or 1=2" pwd='' is entered as input value)
The SQL command firstly, is decomposed by token analyzer into tokens consist of command–based elements, operands, functions and values. The analyzer then applies a policy-based push-and–pop algorithm to initiate the automaton. The state transfer can only happen when the analyzer read a recognized value according to a general rule of algorithm. The algorithm validates the type, length. Furthermore, the tautology policy can be implemented as special rule.
The design of security components complies with enterprise strategy by using semantic mediation to take the business context into micro processes design; by this means each component has its contract level responsibility in provisioning a subset of service, so does a security component. The validation process relies on a common service to take the event into validation against the input strings. Consequently the malicious data can be detected through data validation tracking.

4. Tracking on a configurable topology

Eigenvalue in attacking matrix - In order to eliminate the risk caused by vulnerability, web application validation against each input string become necessary to remove malicious data. The concerns raised as many service providers do not fix server problem, and client site protection become mandatory [9]. In penetration test, source code is analyzed against known patterns to find vulnerabilities which we called black box. The reason of black box is due to the test result doesn't provide detail information of the vulnerability in terms of location or characteristics. Consequently, it doesn't achieve our goal from identification, validation and tracking perspective.
Based on the foundation of identification and validation it achieved, we explore the flow tracking in a macro view architecture. To perform tracking, firstly let E be the matrix sets of Cartesian space consists of the edge eij from node ni to node nj. Let E further denotes the product of flow analysis, which involved in variant flow topology with fixed eij values until a new configuration is involved. eij has a flip-flop value of either "0" for non-exist, or "1" for exist. Matrix E represents a pre-determined control flow graph, which is independent from those attacking events
In contrary to matrix E, let X denotes the vector from attacking site, and xi corresponds to node i. X from attacking site is independent of flow matrix E, as the factor is under attacker's control and out of configuration. From the eigenvalue base equation $EX = \lambda X$, X represents the eigenvector and λ represents the eigenvalue. Thus, we can

derive the base equation into a full representation as followed.

Among all factors, λ is the weight coefficient or, in security's term, "risk amplifier" of the vector X which affect the attacking strength.

$$\begin{bmatrix} e^1_1 - \lambda & e^1_2 & \dots & e^1_k \\ e^2_1 & e^2_1 - \lambda & \dots & e^2_1 \\ \dots\dots & & & \\ e^k_1 & e^k_2 & \dots & e^k_k - \lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ .. \\ x_k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ .. \\ 0 \end{bmatrix}$$

The configurable topology of E actually can be a λ value affecter from those combinational attacking perspectives. In order to derive a lower value of λ, two approaches were studied as means of risk-eliminators. They are discussed as followed.

Common Component and Dominant Hub - The first approach, we term Common Component Discovery (CCD) which is based on the convergence of node base topology, to consolidate the different nodes, of identical or similar functions, into the simplicity which we term "common components". This reconfigured E graph helps the minimizing of λ value as many "peripheral side-effects" will be diminished.

The second approach is the hub-like node generation, which we term Dominant Hub Identification (DHI). From control's standpoint, if all of the path transverse from node ni, can reach node nj, we term nj the dominant of ni., and nj is the immediate dominant of ni if there is no other dominant can be found between ni and nj. By taking example as followed, node "8" is the dominant and immediate dominant of "1".

branch 1: 1-3-5-2-7-9-8-4 . . .
branch 2: 1-3-5-2-8-4 . . .
branch 3: 1-6-9-8-4 . . .

The objectives of CCD and DHI are try to take the validation result and node identification into the coherence of macro view tracking. So the combination scenarios of vulnerability issues can be investigated at a single point supported by metadata techniques.

Apart from the node tracking and issue detection, CCD and DHI contribute the elimination of attacking risk. Besides, the easy configuration and topology simplicity are all advantages in a component-based design and can be the foundation of service base construction such as service oriented architecture (SOA).

Algorithms - The algorithm for CCD identification cannot be automated at this time by means of model based computing. This implies human effort need be involved in identification. However the visibility of a clear alignment between business context and nodes helps the identification as the validation rule is based on the assumption that a business function should be supported by an unique functional node, and reversely cannot be always true as a node actually can support identical business functions aligned with different business components. CCD consolidation can be the foundation of DHI, as from topology's viewpoint as node configuration is simplified and each node perform unique function.

The discovery of DHI, is based on a peer-to-peer node base configuration, which can be automated through a matrix based algorithm. Firstly, we create a matrix to hold the sequence metadata, and have it stored in repository. Figure 3 gives example of the matrix that when the control flow passing through node of 1-9, we use Boolean (1/0) value to note visiting status for either "visited" or "bypassed". Secondly, we perform dominants identification based on a string-matching algorithm. The algorithm uses a static analysis matrix to map node and flow, and to generate a string code such as "0101101" at run-time. The string match of "0101101" case for node 2, 4 and 7 means "all the visiting flow of node 2 reaches node 4 and 7", and reversely "all flow reaches node 7 passing through node 2 and 4" is always true.

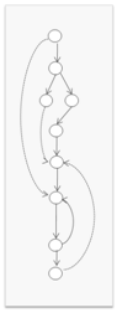
The concept of the algorithm in transverse pattern matching is critical for us to identify DHI, as our vision in static analysis is not limited to the "event based model", instead we add "node centric model" to capture the transversal metadata passing through each node, to complement the weakness from current topologies viewpoint.

FxN	F1	F2	F3	F4	F5	F6	F7
N1	1	0	0	0	1	0	1
N2	0	1	0	1	1	0	1
N3	0	1	1	0	1	1	0
N4	0	1	0	1	1	0	1
N5	1	0	1	0	1	0	0
N6	0	1	1	0	0	0	0
N7	0	1	0	1	1	0	1
N5	1	0	0	0	0	1	0
N9	0	1	0	1	0	0	1

Flow direction

Figure 3. DHI Transverse pattern matching

Transformation - By taking CCD and DHI into a close watch of flow graph for how the transformation being taken place. It is shown in Figure 4, that initially, the general state transfer of base equation can be expressed as followed; $E X = \lambda X$ which should be true for any case.



P2	N1	N2	N3	N4	N5	N6	N7	N8	N9
N1		1					1		
N2			1	1					
N3						1			
N4					1				
N5						1			
N6							1		
N7								1	
N8							1		1
N9						1			

Figure 4 Original topology of node transfer

The figure is a non-DHI case and the node transfer is randomly distributed. The mathematical model of general case can be further derived to the following form;

$$\begin{bmatrix} e^1_1 & e^1_2 & \cdots & e^1_k \\ e^2_1 & e^2 & \cdots & e^2_1 \\ & & \cdots & \\ & & \cdots & \\ e^k_1 & e^k_2 & \cdots & e^k_k \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ .. \\ x_k \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \\ .. \\ x_k \end{bmatrix}$$

Figure 5 gives an example of DHI. Originally the space consists of 9 nodes, and the patterns of CCD and DHI firstly conduct the flow domain decomposition, so the domain will be split into more than one area. Through CCD and DHI processes, the domain is separated into two areas (E_1 *and* E_2), and both areas can be viewed as independent from tracking perspective.

P2	N1	N2	N3	N4	N5	N6	N7	N8	N9
N1		1							
N2			1						
N3					1				
N4									
N5					1				
N6						1		1	
N7									
N8									1
N9									

Figure 5 DHI-type topology for node transfer

The mathematical model of E1 and E2 derived from DHI cases can be expressed as followed;

$$\begin{bmatrix} e^1_1 & e^1_2 & e^1_3 & e^1_4 & e^1_5 \\ e^2_1 & e^2_2 & e^2_3 & e^2_4 & e^2_5 \\ e^3_1 & e^3_2 & e^3_3 & e^3_4 & e^3_5 \\ e^4_1 & e^4_2 & e^4_3 & e^4_4 & e^4_5 \\ e^5_1 & e^5_2 & e^5_3 & e^5_4 & e^5_5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \lambda_1 \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

$$\begin{bmatrix} e^6_1 & e^6_2 & e^6_3 & e^6_4 \\ e^7_1 & e^7_2 & e^7_3 & e^7_4 \\ e^8_1 & e^8_2 & e^8_3 & e^8_4 \\ e^9_1 & e^9_2 & e^9_3 & e^9_4 \end{bmatrix} \begin{bmatrix} x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} = \lambda_2 \begin{bmatrix} x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix}$$

Let $E^* X = E_1 X \cup E_2 X$ be a DHI case, and $X = X_1 \cup X_2$ be always true for any case. In a DHI case; a generic eigen base equation can be expressed as

$$EX=(E_1 \cup E_2)\,(X_1 \cup X_2)=(E_1X_1 \cup E_2X_2) \cup (E_1X_2 \cup E_2X_1)$$

and the eigen characteristics of $E X = \lambda X$ applied for both $E_1 X_1 = \lambda_1 X_1$ *and* $E_2 X_2 = \lambda_2 X_2$

Let "ψ" represents $(E_1 X_2 \cup E_2 X_1)$ which is the peripheral residue of E. This gives the following equation: $EX = (\lambda_1 X_1 \cup \lambda_2 X_2) \cup (\psi) = (E_1 X_1 \cup E_2 X_2) \cup (\psi) = \lambda X$ as a general case, and can be applied to DHC case. In a DHC, as the vulnerability can be easily detected at hub, thus the coefficient λ_1 doesn't affect X_2 deriving $\lambda_1 X_2 = 0$, and similarity applied to $\lambda_2 X_1$. Thus for DHC, the equation can be simplified to an expression of $E^* X = (\lambda_1 + \lambda_2) X \cup (\psi) = \lambda X$ when $(\lambda_1 X_1 \cup \lambda_2 X_2)$ is nullified.

We therefore derive ($\lambda \geq \lambda_1 + \lambda_2$) and further derive ($\lambda \geq \lambda_1$ and $\lambda \geq \lambda_2$) as $\lambda \geq 0$ is always true, and this applied to an arbitrary coefficient, or minimum value " $\lambda_{\min}$ ". This implies the attacker's weight coefficient against each node diminished in DHI effect, and the security risk can be eliminated under such circumstances.

As mentioned previously, in addition to web security, CCD and DHI achieve additional advantages from service based design and can be the portfolio foundation of SOA. The main reasons are that CCD and DHI contribute the simplicity of component base design in achieving advantages of "loosely-coupled" and "reusable". It also helps a easy-routing process configuration from hub-like topology perspective.

5. Architecture review

Architecture components - As it is stated in the introduction, this paper mainly conduct an overview of architectural strategy in reviewing our recent implementation. From the architecture viewpoints, we introduced the macro and micro views. Our research

applied the mediation services to build up the security framework, the metadata strategy and the interoperability between nodes, event and state. In the following paragraphs, we use the terminologies which have been defined in previous research papers, we like to give brief summaries before going further the details of the elements.

System Architecture in three-sites (Client, Front-end Server and Back-end Server)

- Client: Web Browser and execution
- Front-end: Flow Control and Event handling
- Back-end: Event validation, Repository

Architecture View (Macro and Micro view)

- Macro view (Node based transmission): System and process level view of architecture
  - Abstract Syntax Tree (AST)
  - Control Flow Graph (CFG)
  - Data Flow Analyzer (DFA)
- Micro view (Node internal mechanism): Event transaction and system function view of architecture
  - Pattern Matching Techniques (PMT)
  - Lexical Automata Parser (LAP)
  - Validation Event Trigger (VET)

Universal Adaptor (the engine managing codes and syntax in a common way)

- Analyzer - The static analysis uses analyzer to produce abstract syntax tree as a macro view. This support pattern match in syntax level instead of the sequence of the token.
- Event message carrier - The abstract value of a pointer and its attribute, together with the sequence, and flow-sensitive information are capsulated into the event message carrier.
- Pattern match - The tool parses the source codes and try to detect the string match (i.e. strcpy, sprintf). The micro mechanism can dicover vulnerability, with an identification, by using a node pointer as notation of physical code.
- Lexical analysis - Lexical analysis support pattern match in a further step insight of arguments of functions. So each identified node, which match a particular pattern, can be analized for details of the associated arguments.
- Parser - The static analysis is build on both macro and micro mechnism by using parser to produce abstract syntax tree as a macro view. This support pattern match in syntax level instead of the sequence of the token.

Architecture mediator (Interoperability between nodes and between elementse)

- Meta Messaging Carrier (MMC)
- Common Flow Abstract (CFA)
- Finite State Automata (FSA)
- Common Validation Algorithm (CVA)
- Knowledge Base Repository (KBR)

Three layer's interoperability (pointer-values- semantic) - According to our recent research, the static analysis is realized by a pointer-values-semantic (PVS) three-layers interoperability. Pointer is symbolic notation in linkage of symbol and code. Value is the input string pending on validation. Semantic is the node based tracking based on policy. Symbolic pointer means the labeling process by taking the abstract syntax, produced from our syntax parser into a symbolic notation. Patern matching algorithm is based on the finite state automaton (FSA) to perform matching task. Once an input string is sensed, the event trigger initiates the patern matcher and parses through the whole execuatble statement.

The pointer, which maps the physical code and a symbolic identification, carries the metadata messaging across the control flow for tracking purpose. The interference between macro and micro view forms a robust error detection as it contributes the knowledge base for that suspect identification for further analysis. From governance perspective, we applied the semantic repository to maintain the metadata of validation rules, suspicious path, and coherence of security governance for both macro and micro architecture.

This implies the alignment control between physical, logical, operational run-time and semantic is the key in enterprise security transformation. Physical world consists of physical codes, logical terms cover abstract syntax and flow graph, operational run-time means event, while semantics implies the alignment business context, and metadata of logical and physical assets. Macro view is based on the node interconnectivity as atomic level such as state transfer, aggregation and metadata alignment, while micro view require insight such as node definition, component design, metadata format, event validation and Context-Sensitive String Evaluation [10].

Architecture coherence - From system architecture perspective, Figure 6. demonstrates the coherence of our macro and micro view of front end, back end and mediation services.

The mediation of the architecture achieves governance of three-layer (pointer-values- semantic) interoperability. It creates a transparent environment in common practices for that abstract syntax, rule-based arguments validation in automata algorithms, and generic metadata format in flow tracking [11]. Our research distinguished unique viewpoints from the coherence of macro and micro views of integrated services as a whole. Macro view encompasses the flow analysis in node based structure, while micro view works on event analysis such as argument validation [12].

The front end has focus on macro architecture from abstract syntax and node identification perspective, while the back end shed more light on micro view. Among these, the metadata strategy in creating the linkage between physical, logical and semantics has been our creative knowledge contribution in web security. To provide an enterprise framework and to take major language codes into node identification process, we launched universal adaptor engine (UAE) to create a common AST view. The experiments show that our pattern for that individual language by using a universal adaptor to transform the codes into a common AST view, to take key language codes and analyze in a generic syntax format.

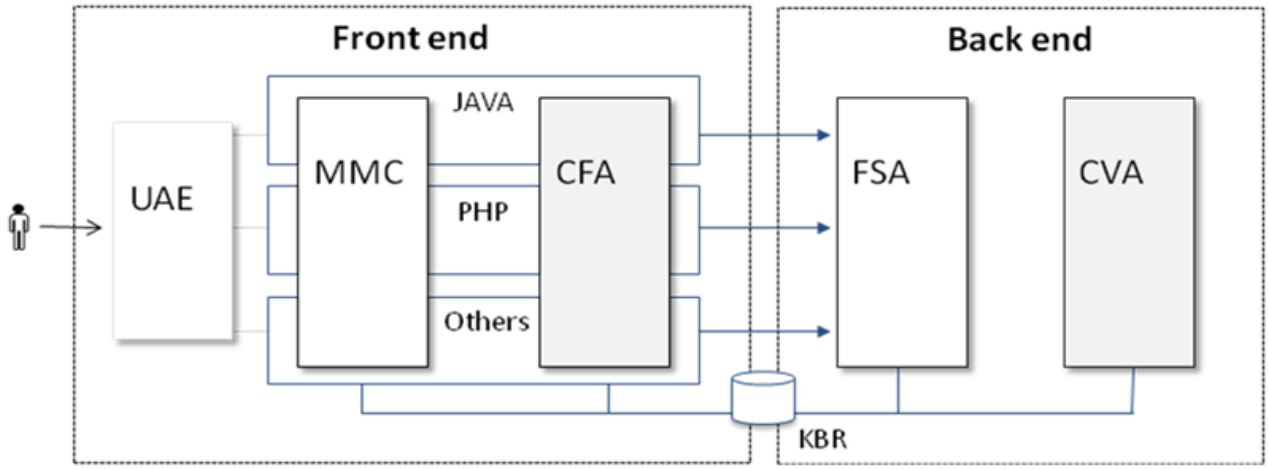


Figure 6. Web Security Architecture Overview

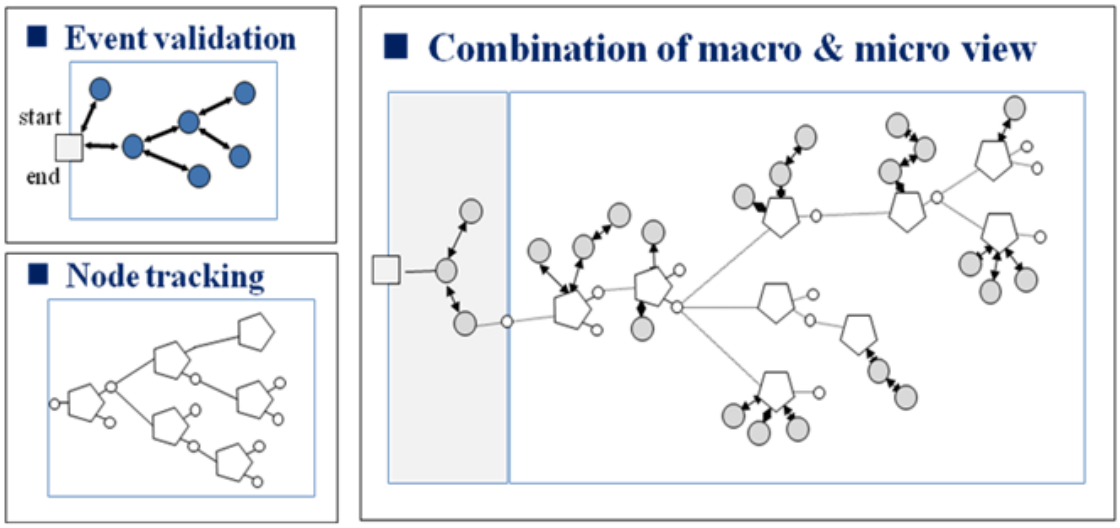


Figure 7 Complementary solution between validation and tracking

To further explain the concept of "the coherence of integrated services", Figure 7 illustrates the complementary solutions of event validation (micro view) and node tracking (macro view) in constructing a healthy architecture. The metadata of node transfer, event state transfer, as well as policy-based validation are stored in the knowledge-based repository. The prediction of tracking is either in static or dynamic form. Static information consists of those static flow control, best practice topology (CCD and DHI recommendations), and suspect node identification. While dynamic information consists of string validation automaton, data transmission tracking over the nodes, and suspect value tracking. Apart from DHI, binding variable is also a key factor in risk minimizing as it eliminate the dependencies in data

transmission and consequently, the sensitivity of vulnerability can be isolated in a fine-tuned topology.

Case study - The proposed PVS architecture can 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.

SELECT <col_1> from $TAB1 WHERE <col_2> = <*inp_1*> and <col_3> = <*inp_2*> ;

inp_1 = ‘ “X”, *inp_2* = ‘101 or 1 = 1 UNION SELECT * from $TAB2; -- ’

inp_1 = “X”, *inp_2* = ‘999; ! HOST INJ= “INSERT INTO USER value (101, “evil”); --“ ! HOST $INJ ’

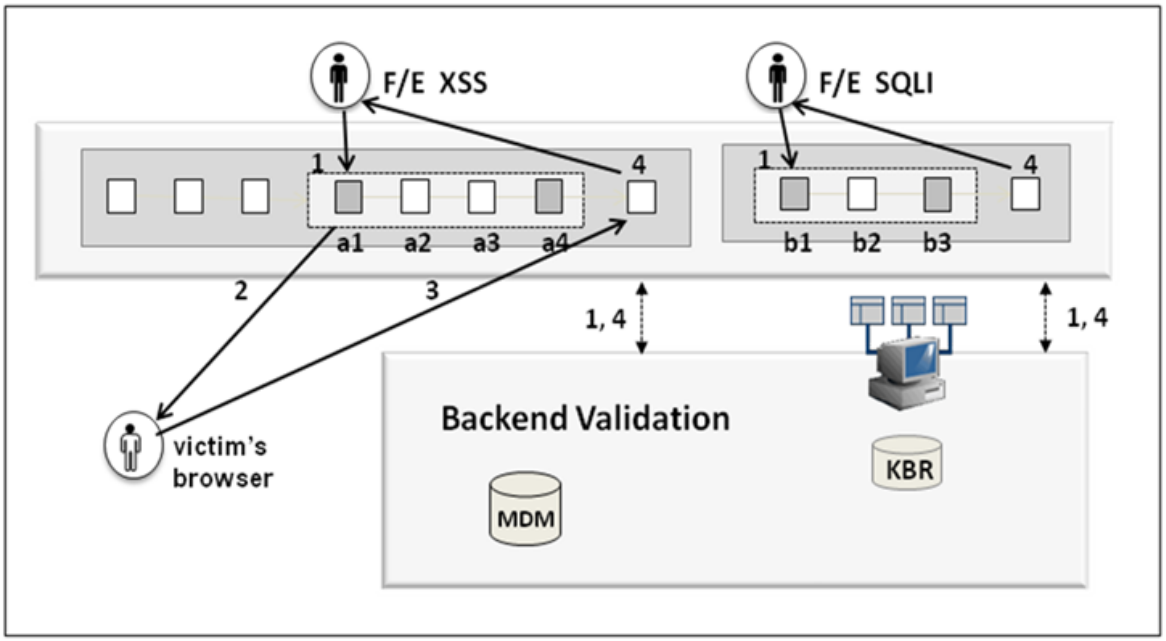


Figure 8 Case study in multi-sites attacking

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 8 illustrates the multi-sites attacking pattern between client, front-end and back-end [13]. The disciplines of vulnerability analysis outline a clear picture of data flow, and mechanism of tracking. However, these may not be sufficient for those sophisticated cases, and further investigation of those combinations of attacking patterns is required.

6. Conclusions

Our research of static analysis shows that the identification of node, by using symbolic notation, is the key to turn black box to gray box, and the initial step for identification was AST definition. To move the abstract syntax structure (AST) and run-time information into encapsulation, we apply metadata messaging to design a pre-defined format as event trigger, and have it embedded to a certain point of the node block. Our system architecture is based on a three-site integration which is client, front-end and back-end. We further apply pointer-value-semantic PVS in explaining the architecture coherence.

The identification process has been successfully implemented and the analysis helps two metadata based branches of research.

The first branch is the validation process, which we term micro view. We apply automata in input string check, so the tokens, decomposed by the analyzer, can be screened by the policy-based engine for a validation. The analyzer,

for producing and analyzing token-base elements, reads the tokens and performs state transfer until the defect or terminator is detected. Defect means the violation of string length, type or tautology regulation. The second branch is node tracking in macro view. Based on the identification process, we propose CCD and DHI models in a configurable topology. The analysis reflects that a component-based common service is the basis in risk minimizing. Furthermore, discovery of the dominant nodes against the configurable topology can be risk-eliminator of web security. For this reason, node tracking, interference between state and event, loosely-coupled architecture, and binding variables are all key factors in our approaches.

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".

References

[1] TCS, IBM and EDS: Abstract Syntax Tree Metamodel (ASTM), OMG Document (2007)

[2] Wu, R., Hisada, M., 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

[3] Xu, W., Bhatkar,S., and Sekar, R.: Practical Dynamic Taint Analysis for Countering Input Validation Attacks on Web Applications, Stony Brook University (2006)

[4] Livshits, B.: Improving Software Security with Precise Static and Runtime Analysis, PhD thesis Stanford University (2006)

[5] Pietraszek1, T., and Berghe, C.: Defending against Injection Attacks through Context-Sensitive String Evaluation, IBM Zurich Research Laboratory and Katholieke Universiteit (2004)

[6] Wassermann, G., and Su, Z.: Static detection of cross-site scripting vulnerabilities, ICSE pp. 171-180 (2008)

[7] Christensen, A., Moller, A., and Schwartzbach, M.: Precise analysis of string expressions, In Proceedings of the 10th Static Analysis Symposium (2003)

[8] Wu, R.: Service design and automata theory, International Conference on Enterprise Information System and Web Technologies (2007)

[9] Shoham S., et al.: Static specification mining using automata-based abstractions, ISSTA pp. 174-184 (2007)

[10] Pietraszek, T., and Berghe, C.: Defending against injection attacks through context sensitive string evaluation, Recent Advances in Intrusion Detection (2005)

[11] Gould, C., Su, Z., and Devanbu, P.: Static checking of dynamically generated queries in database applications, In Proceedings of the 26th International Conference on Software Engineering (2004)

[12] Wu, R., Hisada, M., and Ranaweera, R., 2009, "Static and Dynamic Analysis for Web Security in Generic Format", September 2009, ICGS3 (International Conference on Global Security, Safety and Sustainability) London, UK

[13] Vogt, P. et al. :Cross Site Scripting Prevention with Dynamic Data Tainting and Static Analysis, NDSS (2007) .