Don't Repeat Yourself: Automatically Synthesizing Client-side Validation Code for Web Applications

Nazari Skrupsky Maliheh Monshizadeh Prithvi Bisht Timothy Hinrichs
V.N. Venkatakrishnan Lenore Zuck

Department of Computer Science
University of Illinois at Chicago

Abstract

In this paper, we outline the groundwork for a new software development approach where developers author the server-side application logic and rely on tools to automatically synthesize the corresponding client-side application logic. Our approach uses program analysis techniques to extract a logical specification from the server and synthesizes client code from that specification. Our implementation (WAVES) synthesizes interactive client interfaces that include asynchronous callbacks whose performance and coverage rival that of manually written clients, while ensuring that no new security vulnerabilities are introduced.

I. Introduction

Current practices in mainstream web development isolate the construction of the client component of an application from the server component. These practices are a byproduct of the fact that the client component is often written using a different programming language and platform (HTML and JavaScript in a web browser) than the server (e.g., PHP, Java, ASP), therefore necessitating developers with different skill sets. Independent development is problematic when the client and server share application logic. In this paper we are concerned with a specific kind of application logic shared by the client and server: the input validation logic. Performing input validation on the client improves the user experience because of immediate feedback about errors, and if the validation is entirely self-contained on the client, it reduces network and server load. Performing input validation on the *server* is necessary for security, since otherwise a malicious user can bypass the client validation and supply invalid data to the server [2]. Necessarily then the client and the server must implement the same input validation logic if the application is to give users the interactive experience they expect, while ensuring the security of the application.

In this paper, we pursue a new methodology that aims to improve the development process and achieve a higher level of consistency. In our approach, web developers author the server side input validation of a web application, and WAVES automatically synthesize the input validation for the client. If the input validation must change, the developer changes the server-side code and reruns WAVES. Our approach has three benefits:

- *Development Efficiency* Developers no longer repeat themselves— client code is automatically synthesized.
- *Greater Compatibility and Efficiency* The potential for mismatches is reduced because the code synthesizer can generate compatible validation checks for the client, based on the server checks.
- *Improved Security*. Our approach allows the development team to spend more time on the server side component, and encourages the specification of all validation checks in the server code itself.

Our implementation WAVES uses program analysis techniques to automatically extract a logical representation of the input validation checks on the server and then synthesizes efficient client-side input validation routines. Of particular note is that WAVES generates code for validation checks that involve server-side state and utilize asynchronous requests (AJAX) to perform the required validation. Our experience indicates that our approach offers a promising improvement to current mainstream web development practices.

II. Example

Consider a simple application, where a user supplies her user ID and her password twice (for password confirmation). There are three validation checks performed by this application: (i) the characters in user ID belong to a specified character set, (ii) the two supplied passwords match, and (iii) the user ID is available for creating an account (i.e., it is not already taken by another user). The server side of the application implements these checks. Our goal is to *automatically synthesize*

the corresponding client side input validation routines by analyzing the server code.

III. Our Approach

WAVES (Web Application Validation Extraction and Synthesis), incorporates client side validation in applications in the following four conceptually distinct phases.

(1) Server analysis. WAVES first extracts the input validation constraints enforced by the server using dynamic program analysis. The key insight is that when the server is given an input it accepts, the sequence of if-statements it executes contains all the input validation constraints. So after submitting inputs the server accepts and rewriting the if-statements in terms of the original form field inputs, WAVES has a list of potential input validation constraints. It then analyzes each one to determine if it is truly an input validation constraint—one that when falsified causes the server to reject the input. WAVES then identifies which constraints are dependent on the server's environment (the *dynamic* constraints) and which are not (the *static* constraints).

(2) Client-side code generation. Next WAVES synthesizes client-side code to check the extracted constraints each time the user changes the value of a form field. Static constraints can be checked directly by JavaScript code, but dynamic constraints can only be checked by the server. So for each form field, WAVES generates client side code that first checks if any static constraints are violated and if not sends a message via AJAX to the server asking if any of the dynamic constraints are violated.

(3) Server-side code generation. Each AJAX request from the client asks about errors for a single form field, yet the original server code assumes the user has provided data for all form fields; thus, the original server code is inadequate for responding to AJAX requests. To generate the proper server code for AJAX responses, WAVES performs code slicing on the server code to create a stub that checks for errors on a given field.

(4) Integration. After code generation, the client is augmented with event handlers that properly invoke the generated code and inform the user of errors. The server is also augmented, which in our implementation requires simply copying a file with the generated server code into the appropriate web server directory.

IV. Evaluation

We implemented WAVES for web applications written in PHP and clients written in HTML/JavaScript. It builds on Kaluza [4] (an SMT solver), and Pixy [3] (a tool for PHP dependency analysis).

To evaluate our approach we selected one form from each of the three medium to large and popular PHP applications. For each selected form, we first manually analyzed the server-side code and identified the constraints being checked — we call this the "ideal" synthesis and use it to assess the effectiveness of WAVES. For each application, Column 2 of Table I shows the ideal number of constraints (static + dynamic). As shown in the next column, WAVES was able to synthesize over 83% of the constraints identified by the ideal synthesis.

Application	Ideal	WAVES	Existing
B2Evolution	10+1	7+1	0
WeBid	17+8	16+6	0
WebSubRev	5+1	4+1	5+0

TABLE I
WAVES SYNTHESIZED 83% CONSTRAINTS SUCCESSFULLY.

We also compared the code WAVES synthesized with code written manually by application developers. The third application in our test suite, `WebSubRev`, already had client-side validation. For this form, the server-side code checked 6 constraints (Column 1 Table I), and the developer written client-side code checked 5 constraints (all of which were static). WAVES generated 4 static constraints and 1 dynamic constraint, therefore synthesizing 80% of the static constraints and 100% of the dynamic constraints. (The reason WAVES missed one constraint was a limitation of Kaluza.)

We refer the interested reader to a more detailed technical report [1] that provides an in-depth treatment of issues involved in realizing WAVES as well as experimental data created for our tool.

V. Conclusion

The novel approach to developing web applications reported in this paper allows the developer to improve security (without sacrificing client interactivity) by focusing on hardening the server-side input validation. Our experimental results indicate that automated synthesis can result in highly interactive web applications, and the synthesized checks rival human-generated code in coverage and expressiveness.

References

[1] Automatically Synthesizing Client-side Validation Code for Web Applications. http://alcazar.sisl.rites.uic.edu/wavesTR.pdf, 2012.

[2] BISHT, P., HINRICHS, T., SKRUPSKY, N., BOBROWICZ, R., AND VENKATAKRISHNAN, V. N. NoTamper: Automatic black-box detection of parameter tampering attacks on web applications. In *the 18th ACM Conference on Computer and Communications Security* (Oct. 2010).

[3] JOVANOVIC, N., KRUEGEL, C., AND KIRDA, E. Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities. In *the 27th IEEE Symposium on Security & Privacy* (2006).

[4] SAXENA, P., AKHAWE, D., HANNA, S., MAO, F., MCCAMANT, S., AND SONG, D. A Symbolic Execution Framework for JavaScript. In *SP'10: the 31st IEEE Symposium on Security and Privacy* (2010).