МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ОПТИКИ АТМОСФЕРЫ СО РАН им. В.Е. ЗУЕВА

# НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ИССЛЕДОВАНИИ СЛОЖНЫХ СТРУКТУР

## МАТЕРИАЛЫ
## ДВЕНАДЦАТОЙ КОНФЕРЕНЦИИ С МЕЖДУНАРОДНЫМ УЧАСТИЕМ
## 4–8 июня 2018 г.

*Мероприятие проведено при финансовой поддержке*
*Российского фонда фундаментальных исследований (проект № 18-07-20033)*

Томск
Издательский Дом Томского государственного университета
2018

# TOWARDS CHECKING WEB-SERVICES SECURITY: USING AUTOMATA EQUATIONS AND INEQUALITIES*

*N. Shabaldina*[1], *N. Yevtushenko*[1], *F. Yu*[2]

[1] Tomsk State University, Tomsk, Russia
[2] National Chengchi University, Taipei, Taiwan
NataliaMailBox@mail.ru, nyevtush@gmail.com, yuf918@gmail.com

A lot of works are devoted to security checking of web-services. In our previous papers, we considered various topics in this area, such as approaches for filter design [1] and sanitization [2]. In [3], we started to analyze how automata equations can be used to describe malicious user inputs. In this work, we join our efforts in order to combine the theory of solving automata equations with practical results on web-service security checking.

Given a PHP program for which we would like to describe malicious user inputs that later should be sanitized, a dependency graph for this program can be constructed. The next step is to derive automata that are associated with nodes in the graph, particularly for the sink node that usually represents a sensitive function in the web application. The intersection $S$ of this automaton with the automaton that represents attack patterns allows drawing a conclusion whether a vulnerability can be exploited. The latter happens when the intersection is not empty [1, 2]. In this case, the sanitization is needed for the given description of malicious user inputs. In this joint work, we discuss some approaches to constructing characteristics of malicious user inputs.

In [1], it is suggested to use functions *PreConcatPrefix* and *PreConcatSuffix* for constructing automata that characterize all potential malicious user inputs through the backward symbolic analysis. Given the set $A$ of some fixed sequences, the function *PreConcatSuffix* allows to find all sequences $\beta$ such that the concatenation with at least one sequence of the set $A$ is forbidden: $\exists\ \alpha \in A$: $\alpha\beta \in S$. Correspondingly, given another set $B$ of sequences, the function *PreConcatPrefix* allows to find all sequences $\alpha$ such that the concatenation with at least one of the sequence in set $B$ is forbidden: $\exists\ \beta \in B$: $\alpha\beta \in S$. This approach was implemented in the tool Stranger [4] for vulnerability analysis. However, in fact, this approach can be undoubtedly applied when set $A$ (or $B$) corresponds to one node in the dependency graph, i.e. contains a single sequence. When set $A$ (or $B$) has several sequences we get false alarms as we miss the important information, in particular, for which $\alpha$ ($\beta$) this $\beta$ ($\alpha$) can lead to an attack. As the consequence, a user can be rather unhappy after superfluous sanitization.

In automata notations, the function *PreConcatPrefix* corresponds to the *right quotient* (the algorithm of deriving right quotient is described in [5]); similarly, the function *PreConcatSuffix* corresponds to the *left quotient*. The right quotient is the solution for the inequality $X.B \geq S$, while the left quotient is the solution for the inequality $A.X \geq S$. And both quotients detect dangerous sequences but can also provide false alarms.

For another type of inequalities, i.e., $X.B \leq S$ and $A.X \leq S$, false alarms are not possible. However, in this case, we can miss some forbidden (dangerous) sequences, since the maximal solution to the equation $A.X \cong S$ contains every $\beta$ that is forbidden for each $\alpha \in A$.

In this work, we suggest to decide the above problem by modifying the algorithm of finding the left/right quotient in such a way that it can be applied for the given set $A$ (or $B$) that corresponds to more than one node in the dependency graph, i.e., contains more than one sequence. If the automaton that describe $S$ is a tree automaton, then we can store the information of the connection between corresponding sequences $\alpha$ and $\beta$, in order to avoid false alarms and thus, organize the sanitization process in a more refined way. Our next step is to try this approach for real life examples.

**References**

1. *Fang Yu, Muath Alkhalaf, Tevfik Bultan.* Patching Vulnerabilities with Sanitization Synthesis. Proceedings of the 33th International Conference on Software Engineering (ICSE 2011), Honolulu, U.S., May 2011.
2. *Fang Yu, Ching-Yuan Shueh, Chun-Han Lin,Yu-Fang Chen, Bow-Yaw Wang, and Tevfik Bultan*. "Optimal Sanitization Synthesis for Web Application Vulnerability Repair." In the Proceedings of ACM SIGSOFT the 2016 International Symposium on Software Testing and Analysis (ISSTA'16), Saarbruken, Germany, July 2016
3. *A. Kolomeets, N. Shabaldina, E. Darusenkova, N. Yevtushenko*. Using Models of Finite Transition Systems for Checking Web-Service Security Proceedings of the 18th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices. 2017. P. 151–154.
4. https://vlab.cs.ucsb.edu/stranger/
5. *Peter Linz.* An introduction to formal languages and automata. 5th edition. Jones & Bartlett Learning, 2011. 427 p.

* This work is supported by the Russian Science Foundation (RSF), research project № 16-49-03012.