Raising the Bar for Automated Attacks against Web Applications using Software Diversity

Master thesis

Supervisors: Prof. Dr. Alexander Pretschner, Dr. Martin Ochoa, Sebastian Banescu
Email: {alexander.pretschner, martin.ochoa, sebastian.banescu} @ in.tum.de
Phone: +49 89 289 – 17, 314
Starting date: immediately

Context
A large array of free penetration testing tools is nowadays available for automated vulnerability analysis in web applications [1], many of them for free. These tools are employed both by white-hat (ethical) and black-hat hackers (attackers). In some cases, attackers and ethical testers must develop custom scripts to effectively find vulnerabilities for particular web applications.

After a malicious attacker finds a vulnerability, s/he crafts a code snippet or script which is able to automatically exploit that vulnerability. Generally, the more “reusable” the attack script is, the more profitable it is for the attacker. For instance, if the script executed in the browser of any other end-user of the same web application is able to provide an advantage to the attacker (e.g. XSS, CSRF etc.) then the attacker can re-use the same exploit several times. Similarly, if the same vulnerability discovery technique works for a wide range of applications, an attacker can automate large-scale scanning until he finds a vulnerable target.

Goal
The focus of this thesis is to design and implement a collection of software diversity transformations [2], which can be applied to web applications on the server-side and/or the client-side code. Such transformations are not intended to fix software defects. The goal of such transformations is: (1) to hamper automated penetration testing performed by malicious attackers and/or (2) to affect the reusability of attack scripts, in order to prevent large-scale attacks against other end-users.

In the evaluation phase, the software diversity transformations must be applied to one or more popular open-source web applications (e.g. ownCloud, Drupal, Joomla, Wordpress, phpBB, etc.). To evaluate the effectiveness of the implemented diversity transformations, a set of exploit scripts will be crafted, leveraging disclosed vulnerabilities for the chosen web application(s) from the MITRE CVE website. The success of the exploit scripts will be tested against the diversified web-application(s). Moreover, the effectiveness of the implemented software diversity transformations must be measured as their ability to degrade the results obtained using automated vulnerability scanners and pen-testing tools.

Work-plan
1. Develop knowledge of software diversification techniques:
   a. Read references (for instance [2], [3], [4], [5]) and find related work on this topic
   b. Write a state-of-the-art survey on related work, which presents and compares the investigated techniques.
2. Design and implement software diversity transformations for web applications:
   a. Describe in writing each transformation, offering a rationale regarding
which kind of automated attack(s) it would defend against
b. Implement the chosen transformations either directly over the code of the
chosen web application(s) or as a separate tool which can be later applied
to the source code of web applications.
3. Evaluation of the implementation (case-study):
a. Select a set of penetration testing tools and apply them to the original
version of the chosen web application(s) and to the diversified version(s).
Document the results of this experiment by comparing how diversity
affects the performance and results of the tools.
b. Select a set of vulnerabilities for the chosen web application(s) and
develop scripts to exploit these vulnerabilities. The generation process of
the exploit scripts must assume that the type of diversity transformations
applied to the web application(s), are known to the attacker. The only
unknown to the attacker are the parameters of the diversity
transformations.
c. Measure the effectiveness of the implemented diversity transformations
against the set of exploit scripts.
d. Measure performance impact of the implementation on the target
application.
e. Analyze and discuss security versus performance issues of the
implemented solution.
f. Evaluate difficulty of alternative automated attacks (for instance with
custom scripts)
4. The final thesis document must contain:
a. Description of the problem and motivation for the chosen approach
b. State of the art survey, including analysis of security and performance
c. Rationale for choosing certain technique(s) for implementation
d. Implementation description
e. Evaluation of implementation
f. Discussion on potential security and performance threats
g. Conclusions and future work.

Deliverables
• Virtual machine able to run a demo of the implementation, including instructions
  on how to run the demo.
• The VM should also include the source code of the implementation.
• Technical report with comprehensive documentation of the implementation, i.e.
  design decision, architecture description, API description and usage instructions.
• Final thesis report written in conformance with TUM guidelines.

References
  Information Security.” In Workshop on the Economics of Information Security
  (WEIS), Harvard University, Cambridge, MA, 2005.
[4]. Cox, Benjamin, David Evans, Adrian Filipi, Jonathan Rowanhill, Wei Hu, Jack
  Davidson, John Knight, Anh Nguyen-Tuong, and Jason Hiser. N-Variant Systems:
  Information Center, 2006.
[5]. Forrest, Stephanie, Anil Somayaji, and David H. Ackley. “Building Diverse
  Computer Systems.” In Operating Systems, 1997., The Sixth Workshop on Hot