37

Non-destructive Security Testing of Web Applications

Daniel Dušek

ABSTRACT

Penetration testing of a company's web application on a regular basis comes either with the fear of possible downtime, or with the need for setting up the test environment. This work aims to offer the solution that will both minimize the fear and remove the need for a separate testing environment and will also enable automated regular security scans. This paper presents a new, generally applicable approach to the web application penetration testing that leans heavily towards the automation while at the same time is non-destructive and leaves no permanent traces of the testing on the target application. The proposed approach can serve as a guiding line for implementation of a tool that automates a portion of the standard penetration testing and eases a tester's work. Applicability of this approach is supported through the proof of concept implementation that follows it. A reader of the paper can adapt the proposed approach to developing their own penetration testing tool.

PROPOSED APPROACH

Approach that I am proposing expects URL address to be initially the only input provided to a penetration tester. In the very first phase (OSINT Scan; see Figure 1), a tester should collect as much openly available information as possible about the application residing on provided URL. Collected information should be then filtered, processed (3rd Party Tools and OSINT Report phases; see Figure 1) and categorized. Separate categories are use-case specific and it is up to a penetration tester to properly establish them.

Based on the category to which collected information belongs, additional category-specific processing steps, should be taken (both 3rd Party Tools and OSINT Report phases; see Figure 1).

Finally, a report summarizing the findings will be generated from the results of all the steps taken and provided back to a penetration tester. Depending on the nature of information collected and processed, the output report can be presented as a list of the discovered vulnerabilities, suggested actions or something else entirely.

OSINT Scan Phase — This phase represents the first and the most complex phase of the proposed approach. A tester specifies their areas of interest and requirements, researches and implements tools to scan these areas in an automated manner and runs the scan. This phase is proposed to be split into the following sub-steps:

1. **Define** — Decision what areas of the target application are to be scanned, and how the results are going to be presented.
2. **Research** — Research what tools can be used to fulfill the requirements and to cover the areas specified in the previous step.
3. **Implement** — Implementation and chaining of the tools selected in the previous step. If tester implements their own tool or wrapper around the 3rd party tool, re-usability and automation should be kept in mind.
4. **Run** — Execution of the automated penetration testing. Run output (the artifacts) is passed as an input to the 3rd Party Tools or OSINT Report phase of the approach.

An example of non-parsable artifacts that are fed into the OSINT Report phase is a list of missing security headers of the target page. An example of parsable artifacts fed into the 3rd Party Tools phase for further processing is a list of discovered query string parameters that are candidates to be attacked by reflected XSS detector.

3rd Party Tools Phase — Designed to use of already existing and developed tools for further processing of data or penetration testing tasks. This phase is tightly connected to the previous phase as in the previous phase, wrappers for 3rd party tools are implemented.

Given the example of a list of query strings to be attacked as an input into this phase, the output of this phase takes form of a list of query string parameters vulnerable to the reflected XSS attack.

OSINT Report Phase — Dedicated to working with the acquired artifacts that are not further parsable. The artifacts are acquired from both the OSINT Scan and 3rd Party Tool phases and in this phase are compiled into the final report (the form of this report has been decided in the OSINT Scan phase, Define step.

An output of this phase should be affected by the requirements on the penetration testing as specified by the party that has requested it. In general, the output could take the form of discovered vulnerabilities report, the list of sensitive information or the information and pentesting pathways worth exploring. To illustrate possible output artifacts that are expected to be processed in the final OSINT Report phase: application secrets or access tokens, vulnerable libraries in use, leaking personal information, system users with weak passwords.

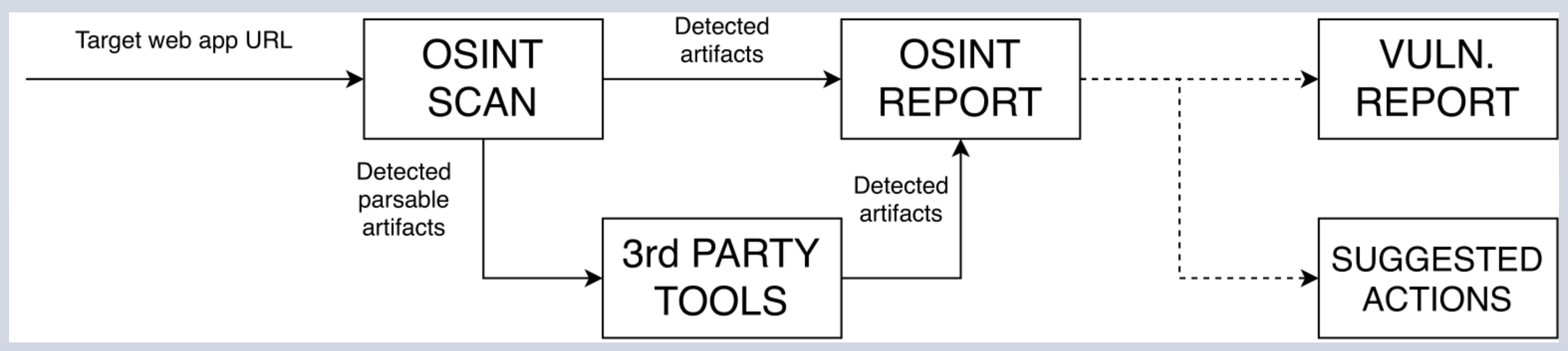

Figure 1: Block schema of the high-level approach proposal. Initially, only the target web application URL is on the input and as the first step, the OSINT Scan phase is initiated. Based on the type of detected artifacts (output) the 3rd Party Tools phase may be initiated to process more complex results. All the detected results that do not require further processing are fed as an input of the OSINT Report phase where final output for a tester is generated. Given by the specific requirements on the testing session, various types of reports may be produced. In the figure, a Vulnerability Report and Suggested Actions reports are used as an example.

CONCLUSIONS

In this paper, I proposed a new approach to web application penetration testing that focuses on non-destructive and non-permanent interactions with the target application. This approach supports automation and when followed, allows for the creation of automated utilities that can be run against the production environment without the fear of breaking it. Later on in the paper, I present the proof of concept tool that I have implemented to demonstrate the possibility of applying proposed principles in practice and to support the approach with an evidence of it.

The implemented tool is capable of penetration testing real production systems without the risk of causing downtime. Furthermore, it automates a lot of repetitive manual tasks that a penetration tester would have to execute and therefore saves time and frees their hands to focus on other, possibly more complex and hard to automate testing tasks.

REFERENCES

[1] Graham Cormode and Balachander Krishnamurthy. Key differences between web 1.0 and web 2.0. First Monday, 13(6), 2008.

[2] Trends in consumer mobility report. Technical Report 1.415.913.4416, Bank of America, August 2017.

[3] Michal Zalewski. The Tangled Web: A Guide to Securing Modern Web Applications. No Starch Press, San Francisco, CA, USA, 1st edition, 2011.

[4] Richard Barber. Hackers profiled—who are they and what are their motivations? Computer Fraud & Security, 2001(2):14–17, 2001.

[5] Andrew Simmonds, Peter Sandilands, and Louis Van Ekert. An ontology for network security attacks. In Asian Applied Computing Conference, pages 317–323. Springer, 2004.

[6] Veronica Valeros. Make it count: an analysis of a brute-forcing botnet. The Journal on Cybercrime & Digital Investigations, 1(1), 2016.

[7] Catalin Cimpanu. Gamarue botnet uses hijacked wordpress sites to send spam with js payloads, April 2016, (accessed November 21, 2018).

[8] Patrick Engebretson. The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy. Syngress Publishing, 2nd edition, 2013.

[9] OWASP. About the open web application security project, September 2018, (accessed November 25, 2018).

[10] A Muller, M Meucci, E Keary, and D Cuthbert. Owasp testing guide 4.0, 2014 (accessed November 21, 2018).

[11] R. Fielding and J. Reschke. Hypertext transfer protocol (http/1.1): Semantics and content. RFC 7231, RFC Editor, June 2014.

[12] Netsparker. Netsparker web application security scanner benefits overview — netsparker, 2019 (accessed April 5, 2019).

[13] Portswigger. Auditing, 2019 (accessed April 5, 2019).

ACKNOWLEDGEMENT & CONTACT

I would like to express my gratitude to my supervisor Ing. Jan Pluskal for his help and guidance through the whole process of working on my master thesis which this article is based on.

Feel the need to tell me something? You can reach me at xdusek21@stud.fit.vutbr.cz until the 17th of June.

Excel@FIT 2019