XIEv: Dynamic Analysis for Crawling and Modeling of Web Applications

Manuel Leithner
SBA Research
Wien, Austria
mleithner@sba-research.org

Dimitris E. Simos
SBA Research
Wien, Austria
dsimos@sba-research.org

ABSTRACT

Researchers and practitioners in the fields of testing, security assessment and web development seeking to evaluate a given web application often have to rely on the existence of a model of the respective system, which is then used as input to task-specific tools. Such models may include information on HTTP endpoints and their parameters, available user actions/event listeners and required assets. Unfortunately, this data is often unavailable in practice, as only rigorous development practices or manual analysis guarantee their existence and correctness. Crawlers based on static analysis have traditionally been used to extract required information from existing sites. Regrettably, these tools can not accurately account for the dynamic behavior introduced by JavaScript and other technologies that are prevalent on modern sites. While methods based on dynamic analysis exist, they are not fully capable of identifying event listeners and their effects. This work presents XIEv, an approach for dynamic analysis of web applications that produces an execution trace usable for the extraction of navigation graphs, identification of bugs at runtime and enumeration of resources requested by each page. It offers improved recognition and selection of event listeners as well as a greater range of observed effects compared to existing approaches.

CCS CONCEPTS

• **Software and its engineering** → **Software verification and validation**; • **Security and privacy** → **Web application security**; • **Information systems** → *Data extraction and integration*; *Service discovery and interfaces.*

KEYWORDS

web crawling, dynamic analysis, modeling, web applications

ACM Reference Format:
Manuel Leithner and Dimitris E. Simos. 2020. XIEv: Dynamic Analysis for Crawling and Modeling of Web Applications. In *The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20), March 30-April 3, 2020, Brno, Czech Republic.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3341105.3373885

SAC '20, March 30-April 3, 2020, Brno, Czech Republic

ACM ISBN 978-1-4503-6866-7/20/03…$15.00
https://doi.org/10.1145/3341105.3373885

1 INTRODUCTION

In the decades since its inception, the internet has rapidly developed to be the most prevalent communication medium for businesses and individuals alike. Web services nowadays are seen as basic building blocks for both commercial and government services. Ensuring the correctness and availability of these platforms is thus a necessity for anyone offering services over the web. A plethora of tools used to assist practitioners in the evaluation of websites exist. These utilities commonly require a comprehensive list of individual pages that make up the target site. Unfortunately, such a list is often not available for a multitude of reasons: Penetration testers try to simulate the experience of an outside attacker and thus view the site as a black box on purpose. Developers seeking to assess the correctness or security posture of their own product generally do have access to the source code, but rising complexity due to increased reliance on external codebases and limitations arising from commercial pressure often lead to a situation where it is infeasible to produce comprehensive and correct documentation on all available pages, REST endpoints, dead functions etc. for all but the most trivial sites.

The traditional (and most commonly used) approach employed by search engines, security assessment and testing tools is to utilize a crawler based on static analysis. Fundamentally, these tools follow a simple process: The HTML source code of a page is parsed, static elements such as images, JavaScript source files and stylesheets are downloaded and all links and form targets are enqueued as additional targets. The procedure terminates when no further targets exist that have not been retrieved and analyzed. Common tools based on this approach include wget [18], Skipfish [17] and w3af [16]. These implementations tend to be very mature and stable; in many cases, they include methods for dealing with issues such as the explosive growth of the number of individual pages due to server-side scripts and localized content. However, most modern sites can not be fully explored purely by analyzing the HTML source. Even disregarding the discrepancies introduced by the choice of parser, dynamic content facilitated by JavaScript, CSS and other technologies modify the behavior of a page in ways that can not feasibly be identified by static analysis. In the most extreme cases (often coined single-page apps), the HTML source of a page consists purely of some static metadata and instructions to load stylesheets and JavaScript code, the latter of which is responsible for fetching and displaying the actual content (which is thus not detected by static analysis). A small number of approaches using dynamic analysis against websites exist in the literature, but they tend to be limited to a small subset of actionable elements on a page, only observe a fraction of possible effects and do not fully take into account the complexity of modern sites.