SOFTWARE SECURITY RESEARCH GROUP
In Collaboration With IBM

Efficient Crawling of Complex Rich Internet Applications

Ali Moosavi, Salman Hooshmand, Gregor v. Bochmann, Guy-Vincent Jourdan, Iosif Viorel Onut

School of Electrical Engineering and Computer Science - University of Ottawa

Introduction – RIAs and Crawling

Crawling is the task of going through all the contents of a web application automatically. Crawlers are used for content indexing, black-box security testing, automated maintenance, etc.

Rich Internet Applications (RIAs) are a new generation of web applications that make heavy use of client side code to present content, using technologies such as **AJAX.** RIAs modify the current page with no need to reload the complete page. This technique has numerous benefits such as better interactivity, reduced traffic, and better responsiveness.

Traditional web crawlers, however, are unable to explore RIAs. The problem of crawling RIAs has been a focus for researchers during recent years, and solutions have been proposed based on constructing a state machine in which:

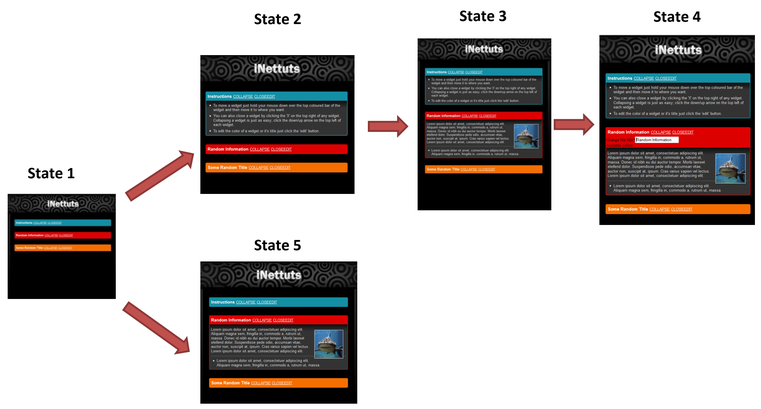
- State = DOM of the webpage
- Transition = JavaScript event execution

In order to ensure coverage, a crawler has to execute all events from all states.

Motivation and Aim

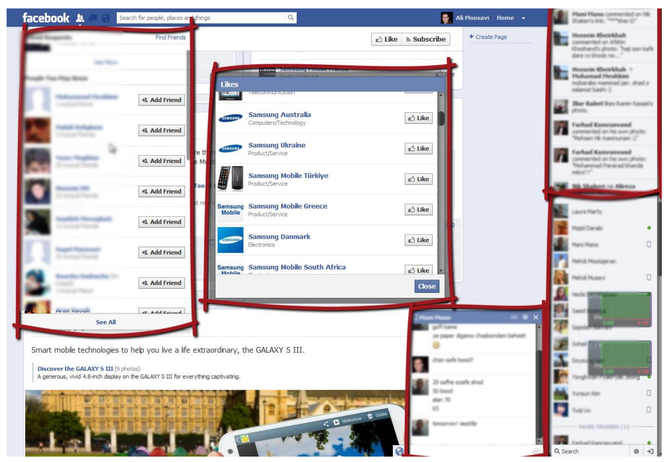
Current DOM-based solutions are only effective on small-scale RIAs. When faced with real-life RIAs, they quickly lose scalability and face **state space explosion.**

The main reason for state space explosion is that different mixtures of the **independent components** of a RIA can produce a great variety of DOMs with no new data.



In the example above, current methods explore the red widget once with the blue widget open (state 3) and once with the blue widget closed (state 5). These two are considered as separate states of the application, and no connection is found between them. This simple website takes hours to be crawled.

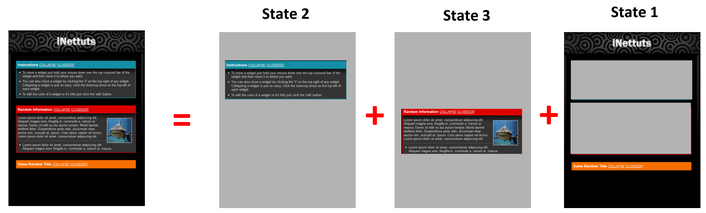
Real-world websites contain many independent components. The picture below shows an example from Facebook.com, with independent components marked with red borders. Current methods are unable to crawl such complex RIAs.



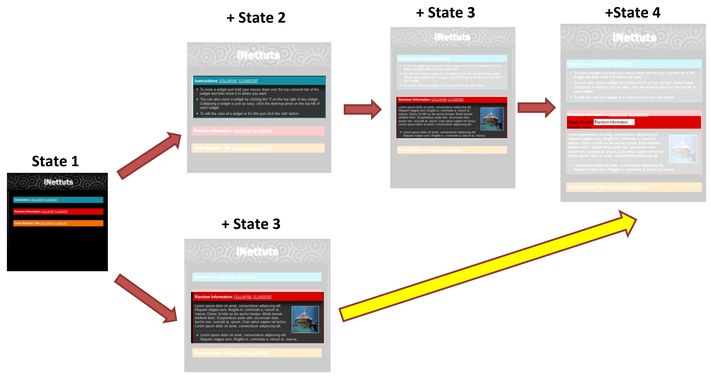
Methodology – Component-based Crawling

Our method models the RIA in terms of components and their individual states, hence it is called **component-based crawling.** In component-based crawling:

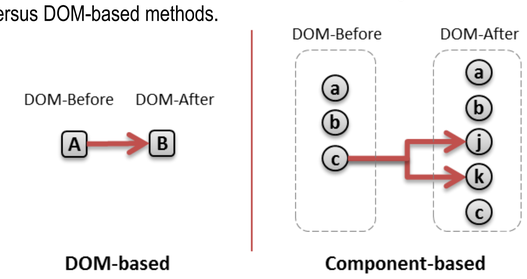
- We partition each DOM to components (subtrees of the HTML tree)
- The current webpage has a ~~state~~ set of states
- States belong to ~~webpages~~ Components



Using component-based crawling, the crawler now knows that the opened red widget is an already-visited state (state 3), and knows how it behaves. (e.g. the yellow transition). Therefore, there is no need to explore this branch further.



In component-based crawling, we model the website as a multi-state-machine.
The multi-state-machine can be represented as a tuple $M = (A, I, \Sigma, \delta)$ where A is the set of states, I is the set of initial states (those that are present in the DOM when the URL is loaded), Σ is the set of events, and δ is a function $A \times \Sigma \rightarrow 2^A$ that defines the set of valid transitions. Picture below illustrates modelling of an event execution in our method versus DOM-based methods.



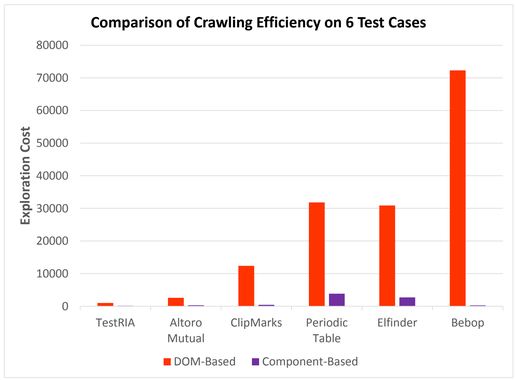
Results

Crawling Efficiency

This chart compares DOM-based and component-based methods on 6 websites, based on the exploration cost, which is:

$$n_e + n_r \times w_r$$

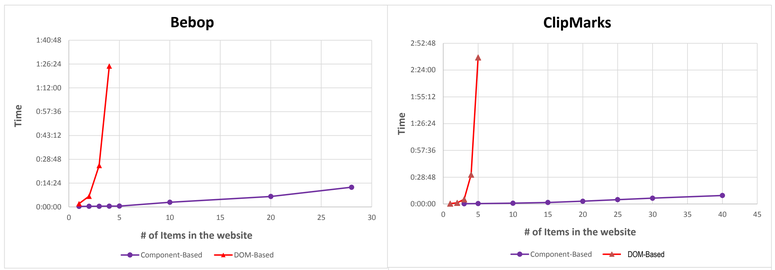
where n_e is the number of events executed, n_r is the number of resets performed, and w_r is the weight of a reset.



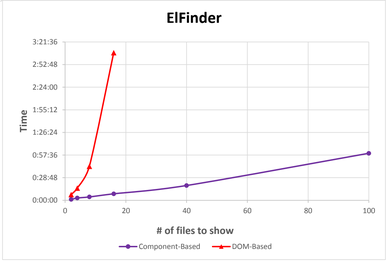
	TestRIA	Altoro Mutual	ClipMarks	Periodic Table	ElFinder	Bebop
DOM-Based	1,003	2,576	12,398	31,814	30,833	72,290
Component-Based	142	308	443	3,856	2,733	293

Results (cont.)

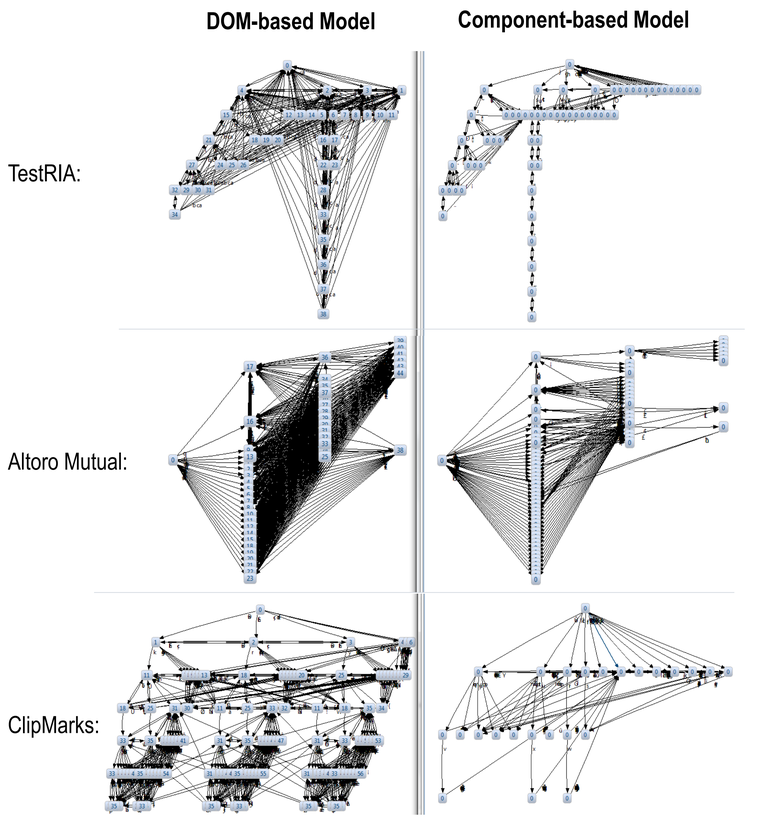
Scalability



These charts show the effect of increasing/decreasing the size of some websites on total crawling time. Where DOM-based crawling time grows exponentially, component-based crawling is able to scale nearly linearly.



Model Size



Component-based crawling yields smaller and simpler models from RIAs, which increases the efficiency of performing security tests using the models.

Conclusions

Component-based crawling provides superior efficiency compared to the current (DOM-based) methods.

- Exponential speed-up in crawling, hence the ability to crawl websites that are not crawlable using DOM-based methods
- Reduced model size, leading to more efficient security testing

Acknowledgments

This work is supported in part by IBM and the Natural Science and Engineering Research Council of Canada.

DISCLAIMER
The views expressed in this poster are the sole responsibility of the authors and do not necessarily reflect those of the Center for Advanced Studies of IBM.