WebRanz Artifact Evaluation

Weihang Wang[1], Yunhui Zheng[2], Xinyu Xing[3], Yonghwi Kwon[1], Xiangyu Zhang[1], Patrick Eugster[1]
[1] Purdue University, USA [2] IBM T.J. Watson Research Center, USA [3] The Pennsylvania State University, USA
{wang1315, kwon58, xyzhang, p}@cs.purdue.edu zhengyu@us.ibm.com uxx16@psu.edu

SUBMISSION URL

https://www.cs.purdue.edu/homes/wang1315/webranz/index.html

LICENSE

- **Makes researchers "smarter" in some way (e.g., identifies and fills some significant gap in prior work)?**
  To the best of our knowledge, this is the first work that aims to understand the ads delivery/blocking mechanisms and introduces counter-measurements.

Figure 1. Screen shots side by side. The red boxes represent the advertisements. The red arrows point the Adblock Plus icons. The gray icon means it is disabled, the red icon means it is enabled and preventing advertisements.

1. OVERVIEW

WebRanz randomizes DOM elements and URLs in order to protect and preserve the web contents from AdBlockers or Web-bots. WebRanz instruments the DOM access APIs of in-page scripts and randomizes the signatures of web elements. The instrumentations are performed on both server side and client side. At the same time, WebRanz preserves the appearances and functionalities of web pages.

Fig. 1 shows three screen shots of an example website without Adblock Plus and with Adblock Plus, as well as the randomized version with Adblock Plus. Our high-level goal is to preserve all the visible elements including advertisements against Adblock Plus by randomizing the web pages.

2. SCORECARD

2.1 Insightful

- **Timely (i.e., addresses a problem that is most current and most pressing)?**
  Yes. Nowadays, a rapidly increasing number of web users are using Ad-blockers to block online advertisements. Ad-blockers could be catastrophic for the economic structure underlying the web, especially considering the rise of Ad blocking as well as the number of technologies and services that rely exclusively on Ads to compensate their cost. WebRanz is a technical approach that allows websites to fend off Ad-blockers and serve their originally-intended Ads.

2.2 Useful

- **Serves a useful purpose?**
  Yes. Our goal is to deliver content publishers' Ads without being blocked and retain a healthy web ecosystem by providing an option for publishers to protect their legitimate rights.
- **Serves a purpose that would otherwise be tedious, prolonged, awkward, or impossible?**
  Yes. Otherwise, the content publisher's ads would be blocked and they will lose money and may not be able to publish free information in the long run
- **Cost-effective?**
  Yes. As we shown in the evaluation, the overhead is low. On the other hand, unblocked ads would bring in more money for the content publishers.

2.3 Usable

- **Easy to understand?** Yes.
- **Accompanied by tutorial notes?** Yes.
- **Artifacts need not be executable but if so, are they:**
  - **Easy to download, install, or execute?** Yes. Please see the tutorial.
  - **Available in a virtual machine image?** Yes. Please see the tutorial.
  - **Available online?** Yes. Please see the tutorial.
  - **Supported by configuration management tools to permit easy updates?** No.

Artifact Evaluation README

[233] WebRanz: Web Page Randomization For Better Advertisement Delivery and Web-Bot Prevention

1. INTRODUCTION

WebRanz is a tool for randomizing Web pages for better advertisement delivery and Web-bot prevention. WebRanz instruments the DOM APIs of in-page scripts and randomizes the signatures of web elements. The instrumentations are performed on both server side and client side. At the same time, WebRanz preserves the appearances and functionalities of web pages.

2. DIRECTORY STRUCTURE

We provide a downloadable link[1] here which includes a zip file webranz.zip. The structure of the zip file is as follows:

```
├── requirements.pdf
└── tool
    ├── install.sh
    ├── run_randomizer.sh
    ├── capture_screenshot.sh
    ├── compare_screenshot.sh
    ├── extensions
    ├── bin
    ├── websites
    │   ├── rand
    │   │   └── showcase
    │   ├── ori
    │   │   └── showcase
    │   ├── rand-url.txt
    │   └── ori-url.txt
    ├── screenshot
    ├── proxy
    ├── src
    └── lib
```

- **requirements.pdf**: Required software with installation links (Note that we also provide a VM[1] that contains all required libraries and programs).
- **tool/install.sh**: Install node modules
- **tool/websites/[ori|rand]/showcase/**: The original and randomized version of the websites
- **tool/websites/[ori|rand]-url.txt**: URLs of the original website and the randomized website
- **tool/screenshot/**: Screenshots of the websites. Each web page has three different screenshots. Details could be found in Sec 3.3.
- **tool/run_randomizer.sh**: Randomize sample websites in tool/websites/ori/showcase/
- **tool/[capture|compare]_screenshot.sh**: Capture and compare screenshots
- **tool/extensions**: Firefox Add-on installation file
- **tool/[lib|src|proxy|bin]/**: WebRanz source code and executables

3. Run WebRanz

3.1 Installation

Run the installation script in directory tool/.

```
$ bash install.sh
```

3.1.1 Virtual Machine

We also provide a VM (Ubuntu) and installed all dependencies. Its home directory is identical to the one listed in Sec.2. E.g. **requirements.pdf** can be found at **"~/requirements.pdf"**. The login and password are **webranz** and **webranz**, respectively.

3.2 Run Randomizer

Invoke the randomizer by running the script (`$ bash run_randomizer.sh`). The script randomizes all the websites under website/ori/showcase/ and generates the output in website/rand/showcase/.

An Example. Fig.1 shows the screenshots of www.yahoo.com before and after randomization: A banner ad on the top center is enclosed in a element with id of "ad-north-base", which matches an Adblock Plus filter and thus blocked. After randomization, the id value is encrypted to a random value "3457323009" so that it will not be blocked anymore.

Among 221 web pages we evaluated, we randomly selected 50 web pages and host on our server at http://50.129.86.149:8080. The URLs of the original and randomized versions are in **"tool/websites/[ori|rand]-url.txt"**. We also provide pre-generated screenshots for the web pages in Table 4 of the paper. They can be found on the page here. Please see Sec.3.3 for details.

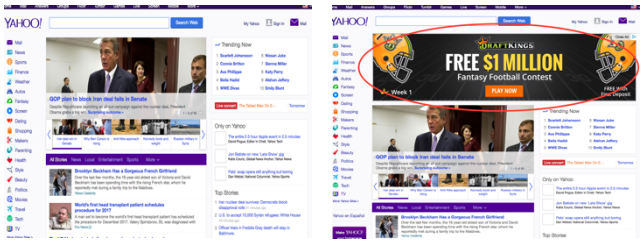

Figure 1. The top banner ad on www.yahoo.com is hidden due to a match on an Adblock Plus filter. Left: original web page. Right: randomized web page. Both with Adblock Plus enabled.

3.3 Capture web page screenshots

To evaluate the effectiveness, for each web *site*, we compare the visual differences among screenshots taken in 3 various settings: (1) ***site-1.png***: the original page without Adblock Plus; (2) ***site-2.png***: the original page with Adblock Plus enabled; (3) ***site-3.png***: randomized page with Adblock Plus enabled.

We prepared a script (`$ bash capture_screenshot.sh`) built atop Selenium to capture above screenshots automatically. It drives the browser, visits URLs listed in tool/websites/ori-url.txt and tool/websites/rand-url.txt, and takes screenshots. Screenshots will be generated and stored in tool/screenshot/.

3.4 Compare Image Difference

The screenshots of the 50 websites can be found in "tool/screenshot". For each site, there are 3 screen shots as described in Section 3.3. **The expected result is that *site*-1.png should be same as or similar to *site*-3.png, while *site*-2.png is different from them as ads are blocked by Adblock Plus.** Invoke the script that compares image differences as follows. `$ bash compare_screenshot.sh`, and the difference score is printed in the console.

[1]https://www.cs.purdue.edu/homes/wang1315/webranz/index.html