Northwestern Lab for Internet and Security Technology

WebShield: A Proxy-Based Mechanism to Detect Malicious Webpages Interactively

Zhichun Li, Yi Tang, Yinzhi Cao, Yan Chen, Bin Liu and Vaibhav Rastogi

1 Introduction

Problem: Malicious webpages (i.e., drive-by-download attacks) that exploit client browsers have become a serious threat. Protecting browsers including legacy ones from vulnerability exploitation is of critical importance.

Design Goals:

1) No browser/host modification

2) High detection accuracy. Ability to detect non-deterministic/event-triggered attacks in dynamic webpages

3) Protect both unknown and known vulnerabilities

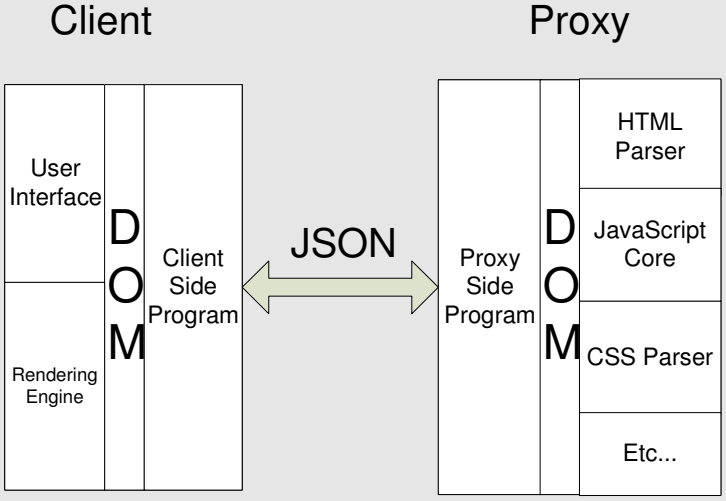
4) Filter attacks and display the webpages with attacks safely as much as possible

State-of-the-art:

- Client side approaches (problem with Goal 1)
- URI labeling: Honeymonkey, Google, Bluecoat and other industrial vendors (problem with Goal 2)
- Proxy-based webpage execution analysis: SpyProxy (problem with Goal 2)
- Webpage rewriting: BrowserShield (problem with Goal 3)

	Browser/host Modification	Detection of Vulnerability in DHTML	Detectionof Unknown Vulnerability
BrowserShield	No	Yes	No
SpyProxy	No	No	Yes
BrowserShield+ SpyProxy	No	When both satisfy, No	
WebShield	**No**	**Yes**	**Yes**

2 System Design



Communication:

Encoding format: JSON

User interaction: Transfer through AJAX

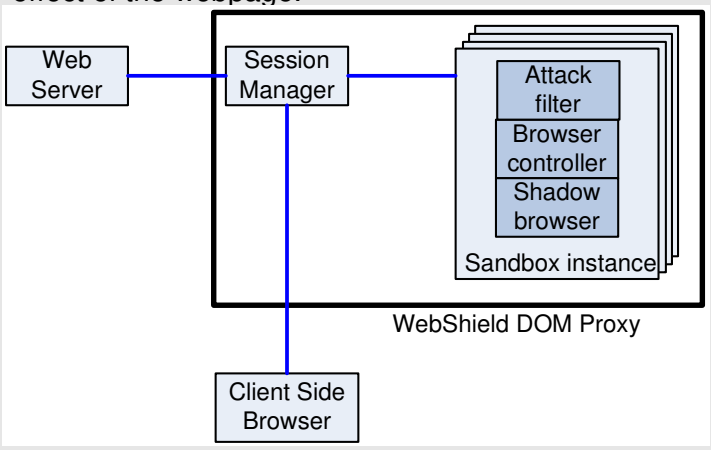
2 System Design (Cont.)

Core Idea: Split browser functionalities into two parts. Run the more dangerous part and detect attacks on the proxy. Only use the client browser to construct DOM, render the visual effects and capture user triggered events. Proxy the DOM updates rather than the HTML pages.

Principle: We want to achieve almost the same visual effect of the original webpage and similar responsiveness of the original webpage.

Proxy Side: 1) Render the webpage (HTML parsing, Javascript running and CSS parsing) and incrementally transform the DOM nodes and CSS stylesheet style objects to the JSON format. 2) Run the policy-based detection for known vulnerabilities and behavior-based detection for unknown vulnerabilities. 3) Upon receiving events from the client side, inject the events into the webpage, and send triggered DOM updates back to the client.

Client Side: 1) Request a URI and download the transformed HTML page including our trusted Javascript code and the JSON transformation of the original webpage. 2) Our Javascript code renders the page incrementally. 3) Our Javascript code captures user inputs and user triggered DOM events, send them back to the proxy side through AJAX call, receive the DOM updates, and update the visual effect of the webpage.



3 Implementation Progress

- ✓ Modify Webkit to implement the Shadow browser
- ✓ SessionManager written in Python
- ✓ Use SELinux for sandboxing (can use Xen also).
- ✓ Behavior based driven-by-download detection. Mechanism similar to SpyProxy
- ✓ Policy based detection engine monitors the DOM interface and other Javascript API calls. Filter the calls that trigger known vulnerabilities.