Proactive and Reactive Mechanisms for Protecting Ads on the Internet from Adware and Malware

Abinash Sarangi
Bing, Artificial Intelligence and Research
Microsoft
Redmond, USA
absarang@microsoft.com

Abstract— **Ads on websites and search engines help keep the Internet services free and accessible to all. These ads are vulnerable to malicious attacks by adware, malware on user's machine and user agent. A webpage has several limitations in its ability to protect its integrity on the user agent. The proposed solution provides the webpage with the ability to incorporate a two-layer protection in preventing malware and restoring the webpage's integrity.**

Keywords-adware prevention; javascript; page validation rules; mutation observer; malicious code injection; malware prevention; Web security.

I. INTRODUCTION

With the Internet's growth, adwares are adopting sophisticated mechanism, running as browser plugins or as background service in order to attack on selective websites. Adware and malware attack is a several millions-dollar industry [2][3] and is being democratized by advanced tools, such as black hole exploit kit [3][4]. A simple adware acting as a browser plugin can manipulate the DOM (Document Object Model) of any website (e.g. Bing or Google) and replace the original ads with its own fraudulent ads. This results in loss of revenue for the affected website. Our research for the work, discussed in this article, suggested at least 4.5% of Bing users had some form of an adware or malware, which inserted unwanted content into Bing's search engine results page, resulting in several million dollars of revenue impact [1][5]. This is true for Google, Facebook and any such Internet service which monetizes using ads [7]. The real challenge of this problem is, the malicious program is running on the user's machine and Internet services are accessed using Web browsers on that machine, limiting what such a service provider can do to prevent the adware from within the webpage itself. This is the very reason why Web services have not been successful in dealing with malware/adwares [5].

The rest of the paper is structured as follows: in Section II, we discuss the problem in detail. In Section III, we discuss the technological and functional aspect of the solution. In Section IV, we discuss the proposed solution, its effectiveness as observed during the research and experimentation. Finally, we conclude the article in Section V.

II. DETAILED PROBLEM DESCRIPTION

Viruses, malwares, adwares and ransomware are getting sophisticated and performing selective attack on websites to generate revenue for themselves by hijacking ads. Search engines and content portals primarily rely on ads on their results page for revenue. When the Web page is loaded on a browser on a machine which is infected with malwares and adwares, the Web page is systematically attacked, and its content is modified. The page's ads are removed, and malware's ads are injected. All of it happens right on the user's browser after the page loads. The search engine or the Web page loses revenue. There is very little the page can do to defend itself on the user's browser. Some important questions to ask when looking for a solution to this problem include: how does the page ensure the page is rendered on the browser as it was emitted by the server? How to ensure the integrity of its content on the client?

Figures 1 and 2 show malware attacks on Bing and Google, respectively.

III. FUNCTIONAL INSTRODUCTION TO SOLUTION

We researched and experimented several mechanisms and built a JavaScript based framework which can consume a rule set generated by server to validate the state of the website and ensure its integrity. The framework, being JavaScript based, can run from within the website seamlessly on all (modern) browsers and devices and protect the webpage from malicious programs running on the user's machine at a higher privilege. The framework uses the unique rule set pertaining to the current page and allows only valid mutation to the DOM from known sources to the website. If any mutation fails the validation, the DOM is restored to the state prior to the mutation. This mechanism ensures that, even if a user's machine has malicious programs or adwares attempting to inject fraudulent ads, the attempt is prevented because the webpage can protect itself from within and ultimately save significant ad revenue. This research has been implemented and thoroughly measured for success and effectiveness through A/B testing at scale [1][5].

Security of websites and services is a growing challenge, and the future Internet needs more research and awareness in the field to deal with any vulnerabilities or exploits that may exist.

 ISBN: 978-1-61208-583-8

IV. SOLUTION

Our research for the work discussed in this article indicates up to 4.3% of search engine users have plugin/malware that modifies the search engine's results page in a way that interferes with the original content [1]. The problem is wide spread in all markets and more on browsers that support plugins or extensions.

A. How does adware work

On an infected machine, the adware either executes as a background process or injects itself as a browser extension. Once active, the adware monitors each browser navigation and executes its checks and monitoring. For example, on Chrome browser, it executes the content scripts to validate if the website being loaded is a target. Once a targeted website e.g. google.com, bing.com. amazon.com etc. is detected, the adware downloads scripts and content (ad images, videos, text, flash objects) specific to the targeted site. The scripts are added to the original page using DOM injection as Script tag. HTML (Hypertext Markup Language) elements are in turn added by the script to the targeted page's DOM. Depending on how severe the interference is, the injected elements could take up part of the site using HTML elements like: DIV, LI, Object, IMG, IFRAME etc. or cover the entire viewport with elements such as modal dialogues. Once elements are injected, the user sees a mixture of content from the original website and the adware. Stolen styles result in deceiving the user to not being able to discern the original content from adware's fraudulent content. Injected ads are contextual and based on the user query as seen in Figures 1 and 2 below.

Usually, the adware code is Polymorphic [8] and encoded e.g:

```
var
l3=(0x12<(0xF5,17.)?'B':(47,22)<=(111.,53.6E1)?(13.83E2,"
a"):(0x1AA,116.7E1));
```

B. Impact of the problem

- Slower Web pages leading to bad user experience
- Ad revenue loss for websites and search engines
- Privacy and security threat for the users
- Unusable website due to clickjacking

C. Proposed solution

The solution that we propose to this problem is a two-step approach. We call the approaches the proactive defense and the reactive defense. Server-side solutions and code have very limited ability as the problem persists on the client side, on the user's machine. We developed a JavaScript based framework which can run on the browser and leverage on the fact that, when the page is generated on the server, we have the knowledge of what the page's content is. The framework utilizes the server generated knowledge to validate the page once it is rendered on the client and either prevents proactively or removes reactively, any anomalies detected.

1. Proactive defense

Adware and malware use the browser APIs using languages such as JavaScript to manipulate the DOM (Document Object Model) of a page. APIs like insertBefore(…), appendChild(…) are used to insert both user interface elements as well as scripts and style elements. In this proposed approach, we override these browser defined functions to user defined functions much before the onload event is fired on the document. In the user-defined avatar of the functions, we validate the element being inserted, with an allowed list or a baseline and either allow or disallow the insertion. For example, if it's a script tag with a source we do not recognize, disallow the insertion. If we know all images on the current page are base64 encoded images, we can disallow all IMG tags with source attribute set to some 3rd party domain. Figure 4 below demonstrates the proactive mechanism with a block diagram.

2. Reactive defense

Our second step of solution is reactive defense. When coupled with the proactive mechanism, the proposed framework provides a two-layer defense, but each of these mechanisms is independent of each other and can be used stand alone. The reactive mechanism in our experiment has proven to be more effective and robust. As the name suggests, this technique requires a rule set generated at the server for the current page. The rule-set is a table that defines the page layout for consumption by the framework, to validate the page and provide a baseline. The rule-set table is generated at server as a map of relative distance of all page components from a static / fixed point on the page, e.g. the search box on a search engine's results page is usually fixed and rest of the content on the page is dynamic, that may change during the page life cycle. So, the map would look something like:

Data-tag: ads_top: [{x:30,y:60,h:80,w:40}, {ads_bgr, img}]

Here, it defines the element with data tag keyword ads_top with its relative position to the fixed point on the page, its dimensions and some metadata attributes, such as any class names and if it has any children element such as IMG, Object etc.

When this map is populated for all the business-critical components of the page, the reactive frame work can add observers on these elements and validate any mutations, i.e. insertion, deletion, style change, visibility change.

Whenever the mutation observer [9] detects a change and the subsequent validation with respect to the rule-set fails, the reactive framework rejects that change and restores the element to the previous state. In our experiment, the 75th percentile performance number for this operation was about 20ms for a series of 20 mutations. Hence, it is not perceived by the user and the page's perceived performance is not impacted. Refer to Figure 3 below for execution steps and flowchart.

 ISBN: 978-1-61208-583-8

D. Experiment details and outcome

A/B testing experiment [10] was done online on a control and treatment group. Treatment and control both had 1 million users, each in markets across en-US, de-DE, en-CA, fr-Fr, en-GB. The experiment was running for a duration of 2 weeks, while revenue and user engagement data were collected. Revenue, as well as user engagement metrics, were statistically significant in positive move in treatment with p-Values in the order of 10^-15.

As much as 0.6% of revenue increase [1] was achieved. In addition, session success rate and click through rates were positively impacted. Overall, it was a successful experiment from a user as well as business value perspective.

E. State of the art review

Existing solutions to mitigate malwares are primarily at operating system and network security level. The user is required to install antimalware applications or turn on security features in the operating system. Antimalware applications rely on a malware and virus signature database which needs constant updates. Malwares can exploit vulnerabilities and affect the user's machine. Another problem with the existing application based solutions is that many users do not have these applications running on their machines. The proposed solution has no action on the user and protects the website from malicious injections from within the Web site. It protects the website (in our research Bing.com) that incorporates this technique and it remains protected even on an infected machine.

V. CONCLUSION AND FUTURE WORK

This paper presented two approaches to protect websites against malwares: Proactive defense, which prevents malicious script injection and Reactive defense, which detects unauthorized change to the website and removes the change to restore the website to its integral state. Though both approaches can work independently, they are most effective when used in combination, resulting in revenue loss prevention and better user experience.

Planned future improvements to this work include making this generic and building this into the browser as a security feature.

ACKNOWLEDGMENT

This work was developed with help from Windows defender research team: Sarvesh Nagpal, Rahul Lal, Marcelo De Barros. Manish Mittal from Bing search engine team helped with research, implementation and experimentation of the solution.

REFERENCES

[1] Windows defender research, "Malware Protection Center", Microsoft Threat Intelligence Journal, pp. 19-21 https://download.microsoft.com/download/D/C/A/DCACBABC-1711-456B-98E1-180E88BFDC68/MMPC%20Threat%20Intelligence%20October%202015.pdf [accessed July 2017]

[2] Sara Yin, "Flashback Malware Robs Google of $10,000/Day in Ad Revenue", PCMag, May 2012 http://securitywatch.pcmag.com/none/297323-flashback-malware-robs-google-of-10-000-day-in-ad-revenue [accessed July 2017]

[3] Technofaq, "An In-depth Look at the Malware Industry", Technofaq.org, May 2015 https://technofaq.org/posts/2015/05/an-in-depth-look-at-the-malware-industry [accessed July 2017]

[4] Jon Oliver, Sandra Cheng, Lala Manly, Joey Zhu, Roland Dela Paz, Sabrina Sioting, "Blackhole Exploit Kit:A Spam Campaign, Not a Series of Individual Spam Runs", Trend Micro Incorporated research paper, 2012, pp. 1-12 https://www.trendmicro.de/cloud-content/us/pdfs/security-intelligence/white-papers/wp_blackhole-exploit-kit.pdf [accessed July 2017]

[5] Ronny Kohavi, Alex Deng, Roger Longbotham, Ya Xu, "Seven Rules of Thumb for Web Site Experimenters", KDD 2014, p. 4 http://www.academia.edu/18352920/Seven_Rules_of_Thumb_for_Web_Site_Experimenters [accessed July 2017]

[6] Robert Siciliano, "Business Identity Theft; Big Brands, Big Problems", HuffingtonPost, Oct 2014 http://www.huffingtonpost.com/robert-siciliano/business-identity-theft-b_b_5643934.html [accessed July 2017]

[7] Larry Dignan, "Google: Click Fraud Costs Us $1 Billion A Year", zdnet, March 2007 http://www.zdnet.com/article/google-click-fraud-costs-us-1-billion-a-year/ [accessed July 2017]

[8] Carey Nachenberg, "Understanding and Managing Polymorphic Viruses", Symantec enterprise papers, volume XXX, pp. 1-4, https://www.symantec.com/avcenter/reference/striker.pdf [accessed July 2017]

[9] Mozilla API Documents, "MutationObservers", July 2017, document explains the API use https://developer.mozilla.org/en-US/docs/Web/API/MutationObserver [accessed July 2017]

[10] Ron Kohavi and Roger Longbotham, "Online Controlled Experiments and A/B Tests", Encyclopedia Of Machine Learning and data Mining, April 2015 http://www.exp-platform.com/Documents/2015%20Online%20Controlled%20Experiments_EncyclopediaOfMLDM.pdf [accessed July 2017]

 ISBN: 978-1-61208-583-8

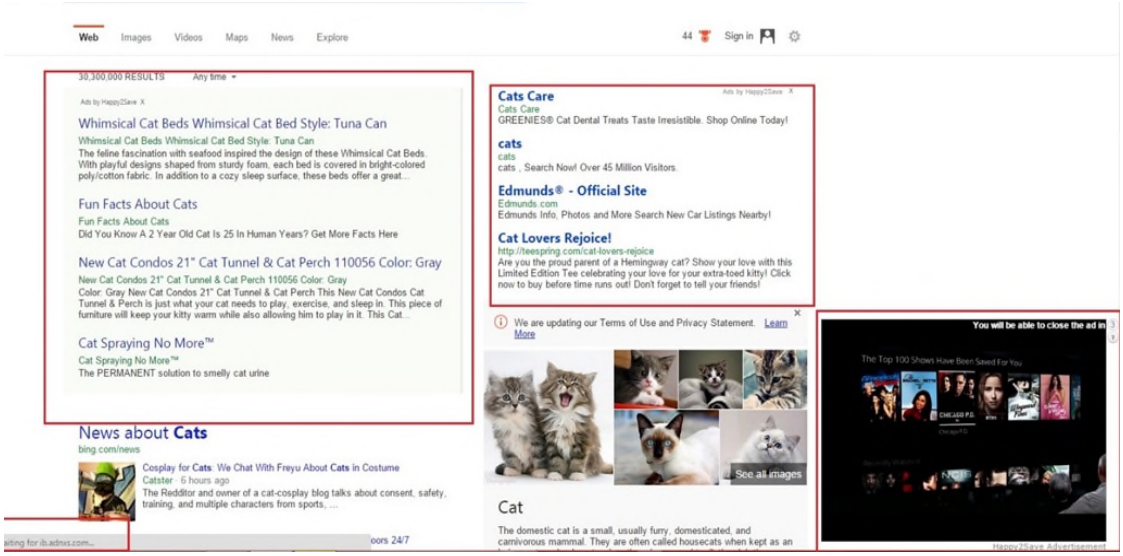

Figure 1. Adware injected ads on Bing's search page. The Ads style (css) mimics that of Bing's styles.

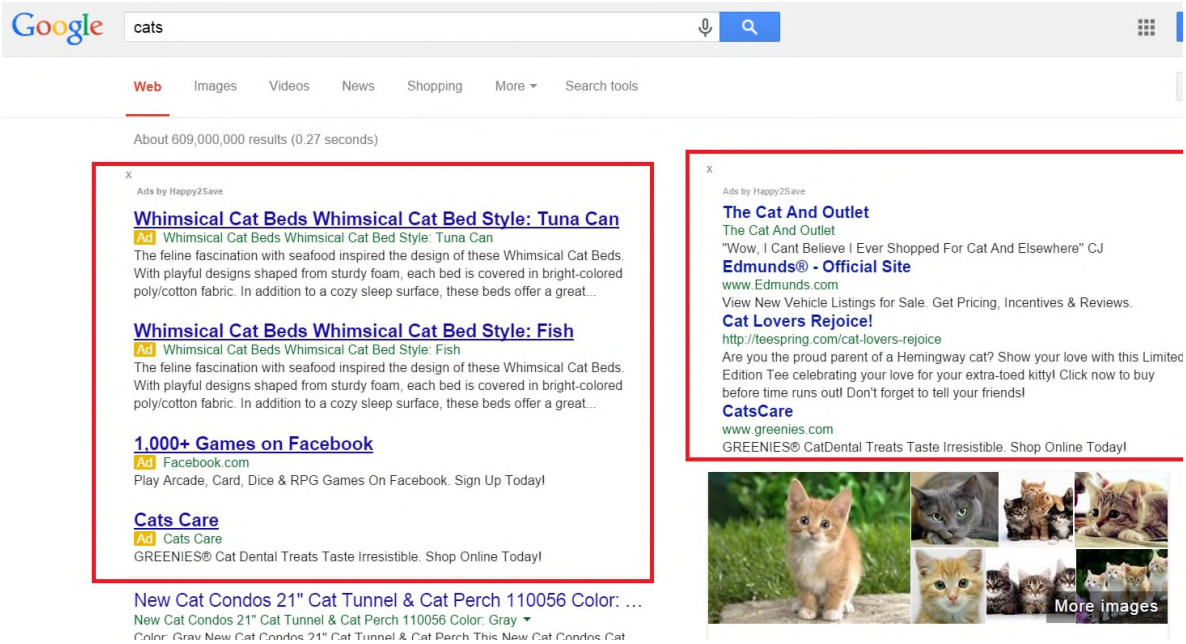

Figure 2. Adware injected ads on Google's search page. The Ads style (css) mimics that of Google's styles.

 ISBN: 978-1-61208-583-8

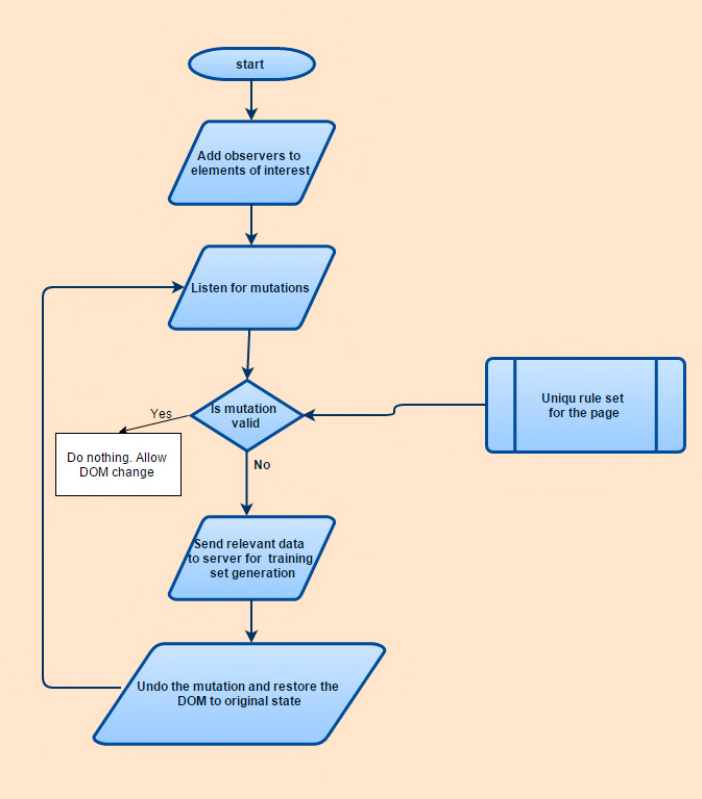

Figure 3: Execution flow of the reactive defense mechanism.

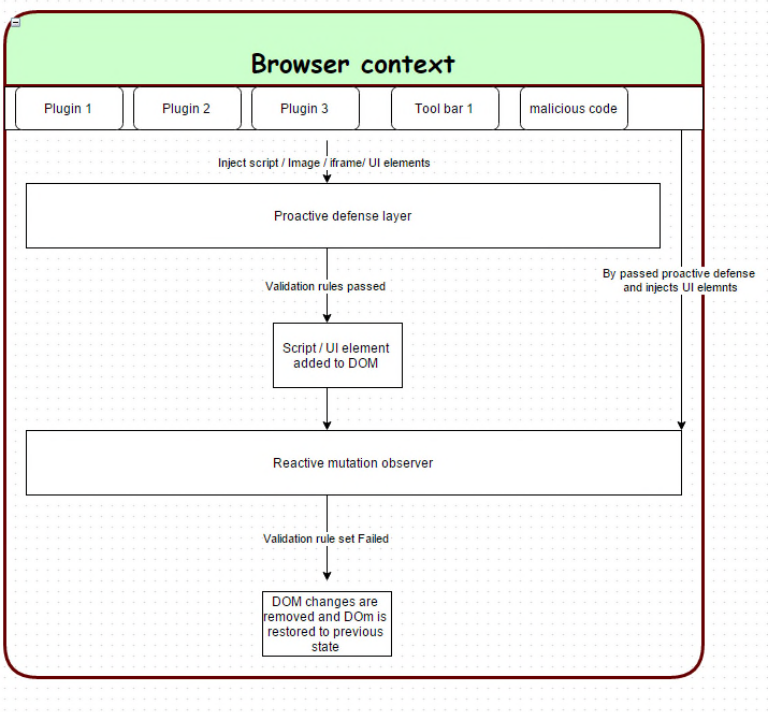

Figure 4: A block diagram explaining where the proactive and reactive defense mechanisms fit in.

 ISBN: 978-1-61208-583-8