Understanding and analyzing threats via malicious browser extensions

Progress Report

In fulfillment of the requirements for the

NU 302 R&D Project

At NIIT University

Submitted by

Kolli Nethre Sai – U101116FCS058

Anshit Saxena - U101116FCS011

Shruti Varghese - U101116FCS123

Rahul Poddar - U101116FCS097

Praneeth Madireddi - U101116FCS09

Abdul Arshiya - U101116FCS001

Area

NIIT University

Neemrana

Rajasthan

CONTENTS

CERTIFICATE

This is to certify that the present research work entitled "Understanding and analyzing threats via malicious browser extensions**" being submitted to NIIT University, Neemrana, Rajasthan, in the fulfillment of the requirements for the course at NIIT University, Neemrana, embodies authentic and faithful record of original research carried out by** Kolli Nethre Sai, Anshit Saxena, Shruti Varghese, Rahul Poddar, Praneeth Madireddi, Abdul Arshiya**, students of B Tech (CSE) at NIIT University, Neemrana. She /He has worked under our supervision and that the matter embodied in this project work has not been submitted, in part or full, as a project report for any course of NIIT University, Neemrana or any other university.**

Name and Title of the Mentor

Dr. Gaurav Varshney

List of Figures

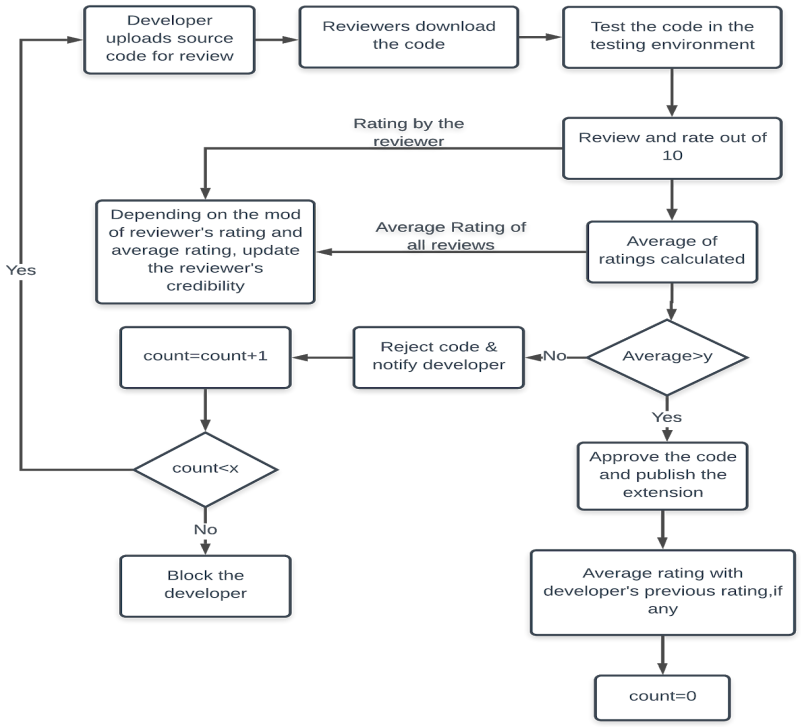


Fig1: Flow chart of Inferno

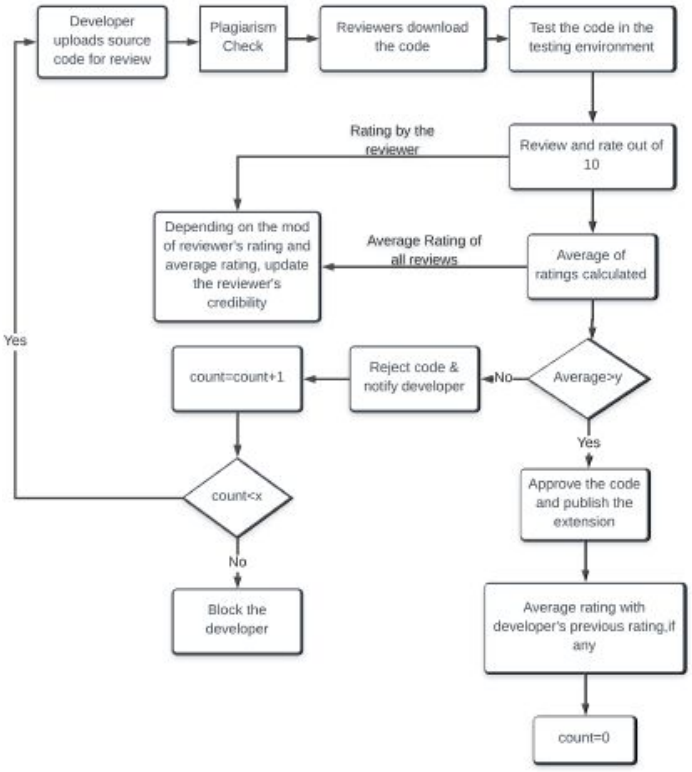


Fig2: Adding plagiarism checker at the beginning

Example:

File A

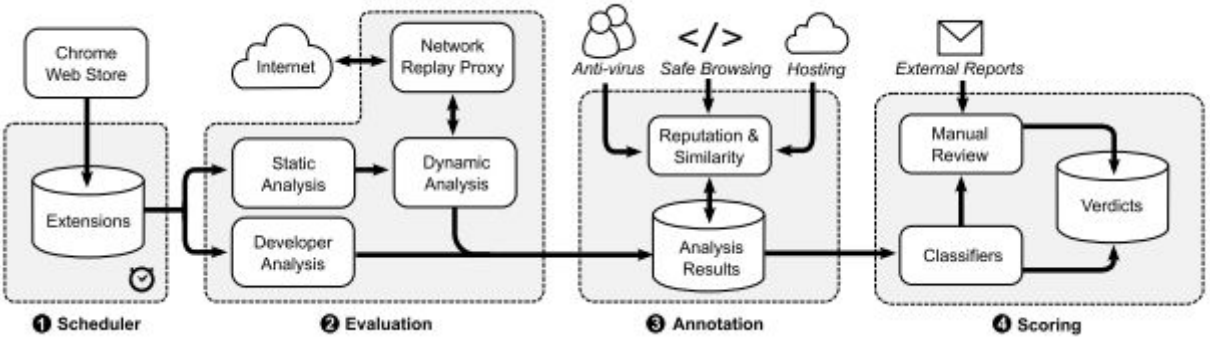
```
if(a < b)
{
    /* Yes! */
    printf("Yes");
}
```

File B

```
if(x < y)
{
    // True
    printf("True");
}
```

A	B	Match
if	if	Yes
(	(	Yes
a	x	No
<	<	Yes
b	y	No
)	)	Yes
{	{	Yes
printf	printf	Yes
(	(	Yes
"Yes"	"True"	No
)	)	Yes
;	;	Yes
}	}	Yes
13	13	3

Fig3: Working of Plagiarism checker



Copied from source:[14]

List of tables

Name of Extension	Extension Description	Attack Type	Attack Description	Browser
HoverZoom	Lets you browse image galleries on various popular websites. (Eg: FB, reddit, etc.)	User tracking/ keylogging	Stores keystrokes by user and the developers have been caught collecting online form data and selling your keystrokes in recent years. In practice they track single webpage you visit and get paid for that data, while simultaneously placing adverts all over the sites you visit most regularly.	Chrome
Mozbar	The all-in-one SEO toolbar for research on the go.	Keylogging	Log user key strokes entered on the browser	Firefox
Fizzle 0.5, 0.5.1, 0.5.2	A RSS/Atom feed using Livemark bookmark feed.	HTML/Web script injection	Javascript code runs in chrome window with chrome developers	Chrome
Social Fixer	Social Fixer for Facebook lets you filter your news feed, hide things you don't want to see, and customize your FB experience.	User Tracking	Its only purpose was to inject code from a remote server that only ran Google Analytics. It was keeping track of which sites users visited.	Chrome
Split Screen	Ot allows users to show two tabs in a single window, while it	Affiliate Fraud		Chrome

	monitors the URLs visited by user			
Musix	Musix is offered as a useful tool with enhancement functions for online music searching.	Affiliate Link	It is an adware that infiltrate the computer by means of third-party program. This is created to generate revenue for its authors.	Chrome
Page Refresh	Allows users to refresh tabs periodically and only requests tabs permissions	cookie stuffing	Uses the background pages to listen all the tab update events that the user visits and thereby stuffs a cookie into users browser.	Chrome
Translate Selection	Recognizes the language of selected text and translates it into the language that was used last time.	Ad Injection	Replaces or Inserts Ads on a web page with the ones provided by the extension.	Chrome
Facebook Rosa	It provides a plugin to view certain videos on Facebook.	Facebook Hijacking	It is a malware and per-forms user actions on Facebook without their consent.	Firefox

Table I: MEs AND THEIR DESCRIPTIONS

Solution	Checking updates/patches
Chrome Web Store	NO
Mozilla	YES
Web Eval [14]	NO
VEX [2]	NO
HULK [8]	NO
LMED [10]	NO
Micro-Privilege Management [13]	YES
Code-integrity Checking Solution [17]	NO
SABRE [7]	YES
STRIDE [15]	NO
Spying Extensions by ML Schemes [3]	NO
LV Detector [16]	NO
INFERNO	YES

Table II.COMPARISONS OF SOLUTIONS ON CHECKING FOR UPDATES/PATCHES OF CODE.

Solution	False alarms for MEs
Chrome Web Store	YES
Mozilla	YES
Web Eval [14]	YES(corrected later)
VEX [2]	YES
HULK [8]	YES
LMED [10]	NO
Micro-Privilege Management [13]	YES
Code-integrity Checking Solution [17]	YES
SABRE [7]	NO
STRIDE [15]	YES
Spying Extensions by ML Schemes [3]	YES
LV Detector [16]	NO
INFERNO	NO

Table III: COMPARISONS OF SOLUTIONS ON FALSE ALARMS FOR ME(s)

Solution	Prevention or Detection
Chrome Web Store	Both
Mozilla	Both
Web Eval [14]	Both
VEX [2]	Detection
HULK [8]	Detection
LMED [10]	Detection
Micro-Privilege Management [13]	Prevention
Code-integrity Checking Solution [17]	Prevention
SABRE [7]	Detection
STRIDE [15]	Detection
Spying Extensions by ML Schemes [3]	Detection
LV Detector [16]	Detection
INFERNO	Prevention

Table IV: COMPARISON OF SOLUTIONS ON PREVENTION OR DETECTION PARAMETER

Solution	Time for Analysis or to Publish
Chrome Web Store	½ - 1 hour
Mozilla	$1hour$
Web Eval [14]	1 hour
VEX [2]	16 sec (flow analysis) / 2 hours (manual analysis)
LMED [10]	53.6 sec
Code-integrity [17]	36.717 sec
INFERNO	3 hours to 3 days (Organization Dependent)

Table V: COMPARISON OF SOLUTIONS ON TIME FOR ANALYSIS OR TO PUBLISH PARAMETER .

The rationale of the work

Browser extensions help users to enhance their browsing experience by providing functionalities like managing passwords and usernames for various sites and increasing the volume. But many extensions can cause harm by being a platform through which phishing, ad injection, DDoS, E-mail spamming and a plethora of other cyber attacks can be executed.

Our research these attacks and the existing solutions to solve them. After analyzing all the solutions, a new solution 'INFERNO' is proposed where reviewers will rate an extension and depending on the ratings, the extension will be deemed safe or unsafe. This solution will help to drastically reduce the number of malicious and buggy extensions uploaded on the browsers.

Literature Review

Introduction

The World Wide Web is a network of online content which include images, videos, text and is identified by a distinct Uniform Resource Locator (URL)[1].The web browsers which are software applications, use these URLs to search in the massive amount of information available on the web. Some examples of browsers being Google Chrome, Mozilla Firefox, Safari. With the internet being available to most of the population, users of these browsers are increasing every day. To enhance an average user's browsing experience, browser extensions are provided.

These extensions provide a range of functionalities like writing JavaScript code on client side [GREASEMONKEY][21], disliking pictures on Facebook [FACEBOOK DISLIKE][22], drawing on live web pages
[WEBPAINT][10]. These attractive extensions are added to Chrome, Safari, Opera, and Firefox as extensions and to Internet Explorer as binary add-ons [10].

But on the flip side, these extensions might cause harm to the users instead of good, if they are malicious.
A malicious browser extension (ME) misuses its privileges and vulnerabilities in the extensions or the underlying architecture of the web browser [23] for the ME

developer's gain. Some of the attacks which can be executed through MEs are listed below in table 1.
Millions of users use browser extensions regularly without realizing their personal information may be in danger or they might be getting scammed through MEs.This poses a threat to an average user's security online.

Many solutions have been developed to detect and prevent Malicious Extensions. All these solutions are described in detail in section. Most of the existing solutions are detection based motivating us to create a prevention-based solution. In this paper, we describe Inferno, an architecture which utilizes static and dynamic information flow analysis to detect and prevent the MEs.The major advantage of the proposed solution, Inferno is its ability to detect all major malicious coding practices possible through a malicious extension and continuous evaluation of the extensions and its updates.Our solution also provides very little false alarms as it depends on the evaluation done by multiple developers.

THREATS

DDoS

A distributed denial of service attack is a malicious attempt of disrupting the normal traffic of a targeted server, service or network by overwhelming the target or its surrounding infrastructure with a flood of internet traffic[12]. MEs can aid in DDoS attacks by sending a huge amount of network packets to a server from the user's browser[9].

Password Sniffing

Sensitive information such as bank accounts and password in transactions made during online shopping is often saved by the web browser. When the victim page is loaded, MEs inject content script into the web page, which can access all the DOM elements including the form with the username and password[13].

E-Mail Spamming

It is also known as Unsolicited Commercial Email (UCE) annoyance to the users. Spammers may purchase a legitimate mailing list or collect e-mail which is posted publicly on the websites. Spam mails can spread malware and are also used to promote products[2]. Email data is stored in rows and MEs can extract data from these through page source or iterate over DOM to get the rows.The extensions can also create pop-ups to get the data[9].

Cross Site Scripting

MEs can inject malicious scripts into websites leading to cross site scripting.This violates the Same Origin Policy(SOP).In 2015, a vulnerability was found in a plugin installed by AVG Antivirus which allows an attacker to perform this attack.Using this plugin, attackers could steal authentication cookies from AVGs website which exposed user's browsing history and other personal information[18].

Facebook Hijacking

The rogue extensions are advertised on Facebook by scammers which claim to do things such as "Change the color of your profile" or "Discover who visited your profile". Once these MEs are installed, it gives the attackers complete control over victim's facebook account, where these accounts are used by the scammers in the business of selling likes to make money[4].

Ad Injection

Ad injection is an activity involving insertion of ads on publisher's web page without taking his/her permission. ME's can insert new ads or replace existing ones during web surfing sessions of a user. Once these MEs are installed, they inject N number of ads on various web pages, which upon clicking ends up in the installation of malware[5].

Open Redirection

An Open Redirection is when a server uses a user-submitted link to redirect the user to a given page or website. MEs can redirect a careful internet user to a site hosting attacker-controlled content, like a browser exploit or a web page executing a CSRF attack(Cross-site request forgery), also known as one click attack[6].

User Session Hijacking

During a session hijacking, a malicious hacker places himself between your computer and website's server(for instance, Facebook), while you are engaged in an active session. At this point, the hacker actively monitors everything that happens on

your account, and even kick you out and take control of it[19]. MEs manipulate the session IDs and session sequence numbers of a website to hack the user's session.

Keylogging

Keylogging is the use of a computer program to record every keystroke made by a computer user, especially in order to gain fraudulent access to passwords and other confidential information. A normal BE having permissions to inject content script to a website or a set of websites can inject a JavaScript code to log user keystrokes entered on the browser.[10] Mozbar[10], Website painter[10], Hoverzoom[9] and Chrome Keylogger[8] are few of the extensions which were detected.

Website Data Manipulation

MEs can manipulate the data of the websites by injecting some adversary which does not take the data but make some subtle, stealthy tweaks to data for some type of gain, can be just as crippling for organizations compared to theft[24].

Affiliate ID Modification

An attacker can use MEs to carry out affiliate fraud with the intention of cheating merchants by deceiving them into paying commissions to them.This is done by changing the tag "my_affiliate_ID to "fraud_affiliate_ID". The extension stealthily tracks websites that the user opens. Whenever the tab is updated, it notifies chrome.tabs.onUpdated.addListener() which adds or changes the affiliate ID in the affiliate link redirection[9].

Stealthy attack via iFrames

In this type of attack, an attacker access pages via extension by loading them into iframes. This attack is one of the noticeable ways in which pages can be loaded in background tabs, or fully transparent or of very small size, rendering them unnoticeable to the user.[11]

History Sniffing

In this type of attack, the extensions take note of all the urls that user is visiting and sent them to the other parties. This can be implemented even with the minimum permissions.[11]

Cookie Stuffing

Background pages are used to listen all the tab update events that the user visits and thereby stuffs a third party cookie into users browser from a website unrelated to that visited by the user.

User Tracking

MEs injects scripts into every website the user visits which allows the hacker to essentially shoulder-surf their visitors by recording and replaying the keystrokes, mouse movements, and scrolling behavior, along with the entire contents of the pages the user visits[20].

Process Monitoring

Recent work showed that real-time network usage data (e.g., packet sizes) can be leveraged to track a user's location and, coupled with knowledge of application behavior or other public information, determine activity within applications (in an extreme case, the content of tweets).[11]

Browshing

The term 'browshing' means phishing through browser extensions.The MEs act as bots on user's machines and a external botmaster controls them through via extension updates.Whenever the user visits the URL which the attacker is interested in, counter is incremented and when the counter passes the threshold value, user is redirected to the phishing website[10].

Existing Solutions

1. Micro-Privilege Management Solution -

Lei et al [13] proposed a micro privilege management solution which separates the privileges of content scripts and extension core and provides a new set of fine grained privileges. This divides the privileges of extension core into two fine grained privileges i.e., inject script and cross site. The content script privileges are divided by sensitivity level, and by new origin and same origin request. Sensitivity level attribute is added by web developers to each sensitive element in the HTML, and browser will

allow the extensions to read the elements whose sensitivity level is same or lower than granted by the user. It is a prevention type solution in which drawbacks are keylogging, affiliate fraud, and browshing.

2. Code Integrity Checking Solution -

Mike et al's [17] focus was on the malicious extensions which are installed on the machine unknowingly through side channels or by malware which can inject their script in the code of the benign extensions. In their proposed technique, the user explicitly sign extension's codes and browser verifies that during the load of extension. Integrity checks FileSigs and CertSigs are added to verify the integrity of extension files and of FileSigs respectively. It is also a prevention type solution in which a drawback is that it cannot prevent attacks that are not generated from extension modification.

3. Lightweight Malicious Extension Detector (LMED) -

Gaurav et al [10] proposed a client side static code analysis based solution named Light weight malicious extension detector (LMED). LMED consists of 4 components, they are Source Code Reader, Static Js Code Analyzer, Malicious Signature Dataset, and Action & Response module. In this, for every N seconds, source code reader reads the Js files related to the extension from the directory and gives it to the static js code analyzer, this component compares the signatures used with the signatures specified in the Malicious signature dataset, if found then Action and Response module reports about that extension as malicious to the webstores. It is a static code analysis detection type solution in which drawback is that it cannot detect the attacks which contain malicious signatures that are not specified in LMED.

4. HULK -

HULK proposed by Kapravelous et al. [8], is a dynamic analysis system which monitors the interaction of Browser extensions with the web pages to discriminate Browser extensions as benign, suspicious or malicious. Hulk uses honey pages to mimic the structure of the web pages that an malicious extension is interested in and uses event handler fuzzing for monitoring and logging the Chrome browser events. But there are some drawbacks in this scheme, the attacks that are not detected by this scheme are Browshing, stealth low privileged masquerading extension.

5. SABRE -

SABRE (Secure Architecture for Browser Extension) proposed by Dhawan and Ganapathy [7], is a dynamic system which monitors JS execution flows to identify malicious extensions. A security label is added with each JS object in its memory with three fields to determine the sensitivity level, its modification status and the list

of JS code from which it is modified. The sensitivity level defines who has authorized access to the object. It also raises an alarm when there is any change in the JS object or any modification is done to the DOM. But it cannot detect attacks which do not include malicious access to JS objects including Browshing.

6. Spying Extensions by applying ML Schemes -

Anupama et al. [3] proposed a feature based detection method of spying malicious extensions by analysis of information such as permissions requested, suspicious file names, spying server URL presence, JS signatures, sensitive information tracking and storage, crowd reputation of Browser extensions, etc. A set of classifiers was trained on the identified features for detection of MEs during validation. The main drawbacks of this scheme is it cannot detect attacks which are not present in the dataset. Affiliate Fraud, Web page manipulation are some attacks which are not detected by this scheme.

7. VEX -

The VEX architecture proposed by Bandhakavi et al. [2] performs the static analysis of JS flows of Browser extensions in a flow sensitive and context sensitive manner. The authors identified that web vulnerabilities can be identified via explicit flows which can be detected. But the drawback of this architecture is that it cannot detect slow paced attacks which do not have explicit JS flows.

8. STRIDE -

STRIDE proposed by Akshay Dev P K and K.P. Jevitha, [15] is used to identify possible threats of chrome specific APIs which are used for extension development. This model maps the possible threats into six categories (Spoofing, Tampering, Information disclosure, Denial of service, elevation of privilege) by analysing the functionality of particular methods, events and properties of each API. But this model did not consider combination of methods and events or two or more methods to identify the threats.

9. LvDetector -

LvDetector proposed by Rui Zhao, Chuan Yue and Qing Yi[16], Combines static and dynamic program analysis for automatic detection of information leakage vulnerabilities in legitimate browser extensions; it aims to be a practical and accurate utility by (1) using a dynamic scenario-driven call graph construction scheme to reduce the overall false positives in the analysis as much as possible, and (2) using static analysis based on each dynamically constructed call graph to extensively analyze the corresponding scenario. LvDetector identified 18 previously unknown

information leakage vulnerabilities in 13 extensions with a 87% accuracy rate. The drawback of this system is it does not aim to be sound at the whole program level.

10. WebEval -

WebEval as described by Jagpal Et al. [14], is a system for detecting malicious browser extensions with a detection rate of 96.5%. The system consists of four parts: Scheduler, Evaluation, Annotation, Scoring. In the scheduler phase either new or updated extensions are uploaded into their system and they analyze it within one hour of submission. Every extension is subjected to evaluation phase that extracts behavioral signals for classification.This includes a reputation scan of the publisher, static analysis of the extension's code, and dynamic analysis that emulates common tasks performed in Chrome. In the Annotation phase they check for the domain blacklists, antivirus engines, and content similarity that contextualizes an extension's behaviors against the larger ecosystem of malicious developers and extensions. In the final step of WebEval returns a verdict for whether to expunge a malicious extension. We use a combination of manually curated rules and a logistic regression classifier re-trained daily over all previously detected malicious extensions to generate a score. A human expert then confirms our automated verdict before passing our decision on to the Chrome Web Store. Every browser extension which is uploaded to the Chrome Webstore is analysed by the system in Scheduling phase.Even revision of the codes triggers a rescan.

Objectives

- To identify the various types of malicious behaviours of browser extensions
- To analyze such malicious behaviours
- To categorise those behaviours by their threat/attack type
- To list out the existing solutions and their efficiency
- Categorizing the existing solutions
- Developing our own solution against malicious browser extension

Methodology

Algorithms

Developer Rating

n = No. of developed extensions
D_{r0} = Developer's initial rating
R = Rating of the current extension

$D_r = ((n * D_{r0}) + R) / (n + 1)$

Weighted Average Calculation

w_i = Credibility score of i^{th} reviewer
x_i = Review ratings of i^{th}
x(bar) = Final review rating for a given extension
FR = x (bar)

$$\overline{x} = \frac{\sum_{i=1}^{n} w_i \cdot x_i}{\sum_{i=1}^{n} w_i} = \frac{w_1 x_1 + w_2 x_2 + ... + w_n x_n}{w_1 + w_2 + ... + w_n}$$

Reviewer Credibility Score

R_0 = Initial credibility rating of the reviewer
FR = Final Rating of the extension
RR = Reviewer's Rating of the extension
n = No. of previous reviews done
x = Reward Range (i.e. FR +- x be the boundaries for constructive rating)

$R = R_0 + [1 - |FR - RR| / x]$ // Instantaneous Rating calculation
$C_r = ((n * R_0) + R) / (n + 1)$ // Credibility Score of reviewer
$n = n + 1$ // Increasing no. of reviews by 1

if($C_r >= 10$)
  $R_0 = 10$

else
  $R_0 = C_r$

Publishing Phase

avg_rate = Resultant rating of the extension.
X = Total no. of developers with ratings >=5

Y = Total no. of developers with ratings <5
R_i = Rating of the i th developer with rating >=5
P_i = Rating of the i th developer with rating <5
avg(p) = Calculates the weighted average of p

```
if( ( avg_rate > avg(Σ_{i=1}^{(α/100)*X} R_i))&&( avg_rate > avg(Σ_{i=1}^{(β/100)*Y} P_i)) )
        return(test pass);
else
        return(test fail);
```

Results of the Work Completed

Inferno detected all possible type of threats while prototyping. No false positives were displayed by the prototype and it had a throughput of approximately accurate results (approx due to test on a small reviewer base).

Analyzing Different Scenarios

1. What is the minimum number of reviewers for analyzing an extension?

   Inferno is flexible, so any organization using this solution can do the following:
   a. Set the minimum number of reviews required (not considering the time)
   b. Set a time frame (not considering the number of reviews.
   For Example,
   a. Org A chooses to put minimum number of reviews as 25% of the total developers
   b. Org B chooses to put a time frame of 4 hours to get the reviews

2. What if a developer steals another developer's work in the name of reviewing?

   By adding a plagiarism checker to the proposed architecture. Example working given below.
   Refer to Fig.2 for Adding Plagiarism Checker
   Refer to Fig.3 for Working of Plagiarism Checker

10 Token of 13 are the same, resulting in a 76.92% similarity. It depends on your individual programming course, if this match count's as equal or not.

3. Is there any way to reduce the time it takes to publish an extension?

Yes! Inferno being flexible can be used with any existing solution. For Example, Web Eval took 1 hour for publishing the extension with one manual review. In this review phase we can add Inferno and now instead of 1 manual review and organization can set a time frame to get multiple reviews from the development community.

Result of Comparisons

Refer to Table II to Table V.

Summary

After doing research on all the possible threats posed by malicious browser extensions and their existing solutions proposed in different research papers, a new solution named INFERNO has been developed by us which acts as a security layer between the developer and the browser. This architecture was designed to prevent malicious browser extensions from entering into the extension store which takes reviews from the developer community and then checks if the extension is fit for the users. Inferno gave positive results on prototyping and being flexible it can be used with any existing solution.

Future Work

The future work of this research will include making Inferno more feasible to the newly created browsers, decreasing the time required for analysing and publishing an extension, Widening the base of this architecture to mobile/desktop application stores, making Inferno more feasible with constricted developer base and associating a reward based review system and also incorporating an in-built plagiarism checker.

REFERENCES

[1] World Wide Web Foundation Available: https://webfoundation.org/

[2] S. Bandhakavi, S. T. King, P. Madhusudan, and M. Winslett, “VEX: Vetting Browser Extensions for Security Vulnerabilities,” *in USENIX Security Symposium*, 2010, pp. 339-354.

[3] A. Aggarwal, S. Kumar, A. Shah, B. Viswanath, L. Zhang, and P. Kumaraguru, “Spying Browser Extensions: Analysis and Detection,” *arXiv preprint arXiv:1612.00766*, 2016.

[4] Emil Protalinski. (2012,March 26). *Malicious chrome extensions hijack facebook accounts.*
Available:
https://www.zdnet.com/article/malicious-chrome-extensions-hijack-facebook-accounts/

[5] Rashmita Behera. (2019). *What is ad injection.*
Available : https://www.adpushup.com/blog/what-is-ad-injection/

[6] Sven Morgenroth. (2018, January 2018). *What is an open redirection vulnerability and how*
Available:
https://dzone.com/articles/what-is-an-open-redirection-vulnerability-and-how

[7] M. Dhawan and V. Ganapathy, “Analyzing information flow in JavaScript-based browser extensions,” *in Computer Security Applications Conference, 2009.* ACSAC'09. Annual, 2009, pp. 382-391.

[8] A. Kapravelos, C. Grier, N. Chachra, C. Kruegel, G. Vigna, and V. Paxson, “Hulk: Eliciting Malicious Behavior in Browser Extensions,” *in USENIX Security,* 2014, pp. 641-654.

[9]G. Varshney, S. Bagade and S. Sinha, "Malicious browser extensions: A growing threat: A case study on Google Chrome: Ongoing work in progress," *2018 International Conference on Information Networking (ICOIN)*, Chiang Mai, 2018, pp. 188-193.

[10]G. Varshney, M. Mishra and P. K. Atrey, “Detecting Spying and Fraud Browser Extensions: Short Paper, ” *MPS ‘17 Proceedings of the 2017 on Multimedia Privacy and Security*, Dallas, Texas, USA - October 30 - 30,2017,pp. 45-52.

[11] L. Bauer, S. Cai, L. Jia, T. Passaro and Y. Tian, "Analyzing the dangers posed by Chrome extensions," *2014 IEEE Conference on Communications and Network Security*, San Francisco, CA, 2014, pp. 184-192.

[12] Cloudfare. *What is a DDOS attack?*
Available: https://www.cloudflare.com/en-in/learning/

[13] L. Liu, X. Zhang, G. Yan, and S. Chen, “Chrome Extensions: Threat Analysis and Countermeasures,” *in NDSS*, 2012.

[14] N. Jagpal, E. Dingle, J.-P. Gravel, P. Mavrommatis, N. Provos, M. A. Rajab, and K. Thomas, “Trends and Lessons from Three Years Fighting Malicious Extensions,” *in USENIX Security*, 2015, pp. 579-593.

[15]P. K. Akshay Dev and K. P. Jevitha, “Stride based analysis of the chrome browser extensions api”, *in Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications* : FICTA 2016, Volume 2, S. C. Satapathy, V. Bhateja, S. K. Udgata, and P. K. Pattnaik, Eds. Singapore: Springer Singapore, 2017, pp. 169–178.

[16]R. Zhao, C. Yue, Q. Yi, Automatic detection of information leakage vulnerabilities in browser extensions, in *Proceedings of the 24th International Conference on World Wide Web* (International World Wide Web Conferences Steering Committee, 2015)

[17] M. Ter Louw, J. S. Lim, and V. N. Venkatakrishnan, “Enhancing web browser security against malware extensions,” Journal in Computer Virology, vol. 4, pp. 179-195, 2008.

[18]Thu Pham. (2016,February 01). *Malicious browser extensions steal user data*
Available: https://duo.com/blog/malicious-browser-extensions-steal-user-data

[19] Paul Cucu. (2017, August 04). *Session hijacking*
Available: https://heimdalsecurity.com/blog/session-hijacking/

[20] Liam Tung. (2018, February 02). *Google chrome beware these malicious extensions that record everything you do*
Available:https://www.zdnet.com/article/google-chrome-beware-these-malicious-extensions-that-record-everything-you-do/

[21] Van Acker, Steven & Nikiforakis, Nick & Desmet, Lieven & Piessens, Frank & Joosen, Wouter. (2014). Monkey-in-the-browser: malware and vulnerabilities in augmented browsing script markets. 10.1145/2590296.2590311.

[22] H. Shahriar, K. Weldemariam, T. Lutellier and M. Zulkernine, "A Model-Based Detection of Vulnerable and Malicious Browser Extensions," *2013 IEEE 7th International Conference on Software Security and Reliability*, Gaithersburg, MD, 2013, pp. 198-207.

[23] I. Sanchez-Rola, I. Santos, and D. Balzarotti, “Extension Breakdown: “ Security Analysis of Browsers Extension Resources Control Policies,” *in USENIX Security Symposium*, 2017, pp. 679–694.

[24] Malware and vulnerabilities. (2019, February 06). *What is a data manipulation attack and how to mitigate against them*
Available:https://cyware.com/news/what-is-a-data-manipulation-attack-and-how-to-mitigate-against-them-97cbe896