Helping or Hindering? How Browser Extensions Undermine Security

Sumonto Chatterjee(2022201028), Rishabh Gupta(2022202011), Soumadeep Acharya(2022201059

under the guide of

.....

Introduction

Prior studies have reported the abuse of extensions to perform nefarious actions such as history-sniffing, data theft, or ad-injection

- in modern day, we use browser extension instead of native web application
- it's more potent because :
  - it improves appearance of web sites
  - allowed third party service.
  - integrate different features using JavaScript APIs.
  - allowed read/write operation in DOM content.
  - and also for controlling client server communication.
- But adding extension may degrade the security. different type of Web application based attack face regularly are:Cross-Site Scripting,framing-based attacks, TLS downgrading , click-jacking attack etc.
- HTTP headers from the server and subsequent client are responsible for delivering different type mitigation technique against of these type of attack.
- Maximum reporting abuse of extensions are history-sniffing, data theft, or ad-injection.

Our challenges

- By using a hybrid type analysis technique , this paper present an automated pipeline , which will detect all the extension that alter security headers.and it gives the security boundary.
- the framework statically observe codebase and analyze runtime behaviour of extension.In this paper , they did a large scale study of chrome web store and detect 1129 extension that alter security headers.
- paper analysis between the three snap- shots of the Chrome extensions . And then it discuss the temporal evolution of the extension ecosystem.
- And for cross verification these framework also applying on firefox . and then it making available for public.

Security Headers

- Security-related Response Headers
  - Content-Security Policy (CSP): To eliminate the dreaded it introduces 'unsafe-inline' keyword, nonces and hashes to selectively allow the inline scripts and 'strict-dynamic' keyword.
  - HTTP Strict-Transport-Security (HSTS): when browser receives this , http header will upgraded to https , and gives a secure channel to prevent diiferent type of man in middle attack like protocol downgrade attacks and cookie hijacking.
  - X-Frame-Options (XFO): it restrict the framing of the website and prevent click-jacking attacks on the client side .
  - X-Content-Type-Options: it instruct the browser to not determine MIME type of content and prevent the application vulnerable to content sniffing.
  - Cross-Origin Resource Sharing (CORS): it instruct browser to make con- tent available to JavaScript even though the content comes from a different origin than the requesting page.
  - Set-Cookie: cookies also carry security attributes. an to mitigate XSS attack it always sends through HTTPS channel .

Security Headers

- Security-related Request Headers
  - Referer and Origin: referer headers sends the entire url to the server of specific documents.this helps server to make a security decision . it is not possible to modified freely if an attacker strip the header
  - Fetch Metadata: Cross-site leaks happens ehen server does not know that who make the request , the origin of request , and the resources. the W3C has proposed Fetch Metadata Request Headers . The idea to indicate to the server the information it actually needs to make security decisions
  - upgrade-insecure-requests: client can informs the server that it wants encrypted channel .And HTTPS is increasingly used with full browser support.

Extension Architecture

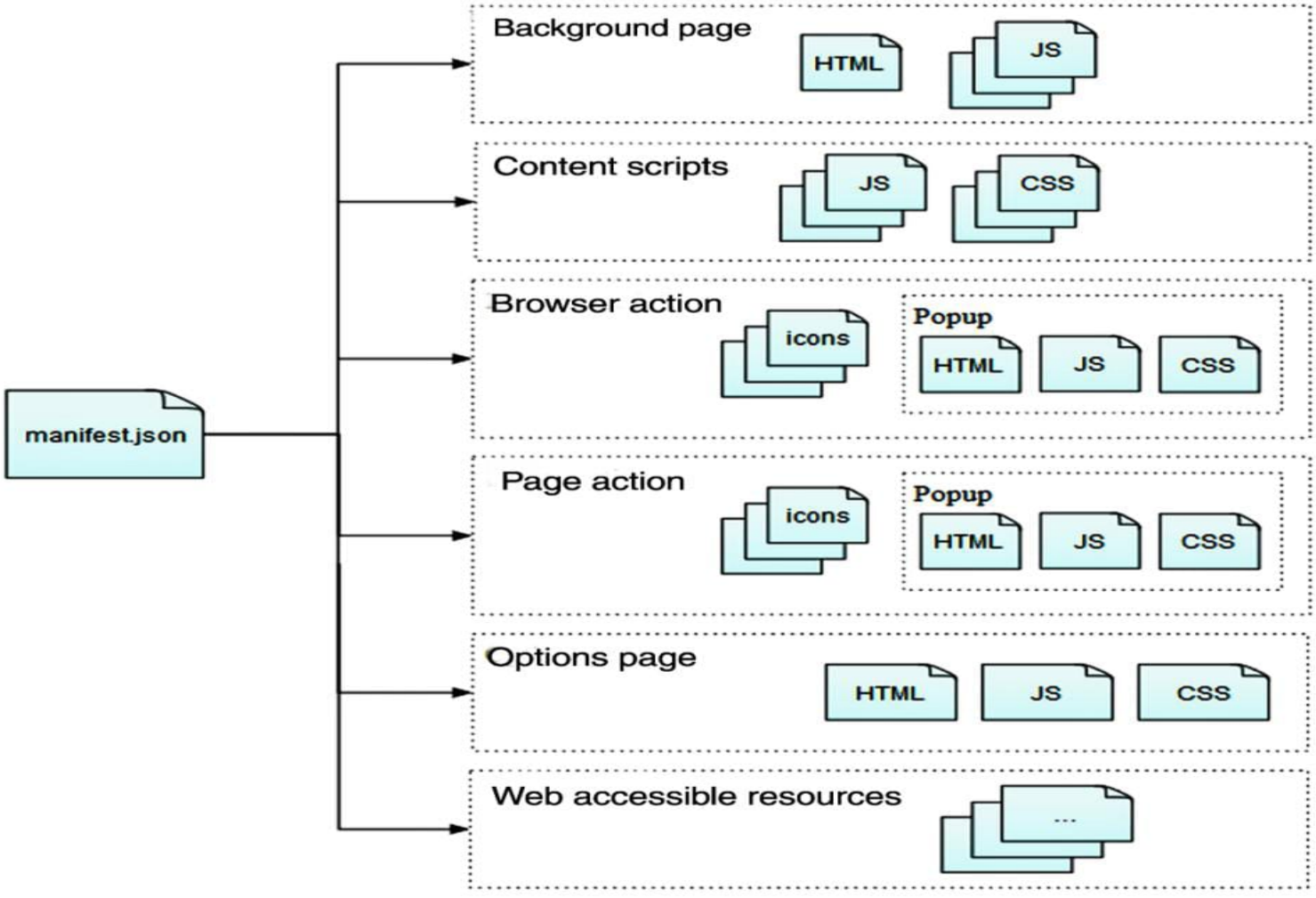

Figure 1: Extension Architecture

Extension Architecture

- manifesto.json This file contains all basics metadata like name, version, and the permissions it requires. It also gives reference to all files in the extension. code:

manifest.json

```
{

 "manifest_version": 2,
 "name": "Borderify",
 "version": "1.0",

 "description": "Adds a red border to all webpages matching mozilla.org.",

 "icons": {
  "48": "icons/border-48.png"
 },

 "permissions": [
  "activeTab",
  "clipboardRead"
  "clipboardWrite"
 ],

 "content_scripts": [
  {
   "matches": ["*://*.mozilla.org/*"],
   "js": ["borderify.js"]
  }
 ]

}
```

RESEARCH METHODOLOGY

in this research paper pipeline use for the following fundamental questions

- in case of intercept or modify Web requests and responses, How many extensions hold the privileges respectively ?
- in case of modify HTTP headers at run-time, How many of the above also utilize the requested capabilities ?
- How many of them actively inject,drop,or overwrite security-related headers on respective target hosts?
- Which security headers are most often altered by these extensions? Do these modifications degrade the client-side security of Web applications in the wild?
- Does this trend of security header interception and alteration among extensions change over time?

RESEARCH METHODOLOGY: Extension Analysis

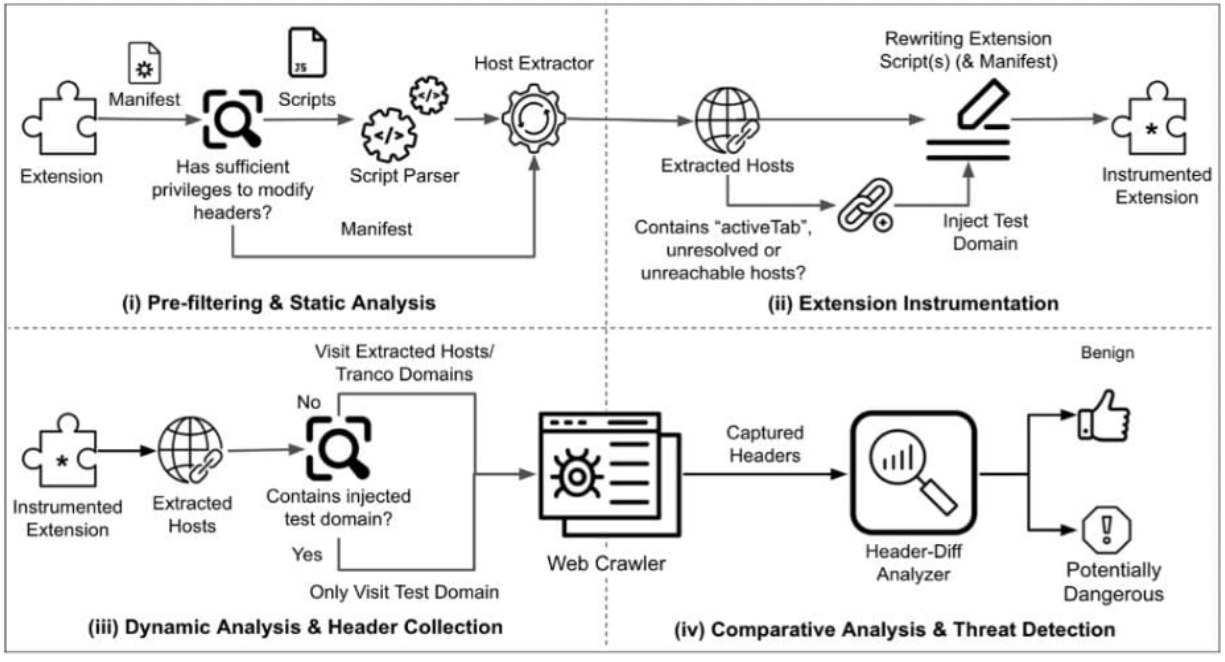

Figure 2: An overview of the automated framework for analysis.

Figure 3:

RESEARCH METHODOLOGY : Extension Analysis

- this framework statically analysis and identify those extensions that can be able to potentially intercept between client-server exchanges
- An extension is useful for both webRequest and the webRequestBlocking permission. it intercept and modify request and response headers synchronously at run-time.
- Using This framework,URL literals is extracted from the code. it falls back to manifest declarations.
- then wildcards will be replaced with valid counterparts . then for dynamic analysis, it stores the target hosts . The framework can also detect extensions that define the target APIs

```
window.browser = window.browser || window.chrome;
var oneMoreDomain = "*://*.bar.com/";
browser.webRequest.onBeforeSendHeaders.addListener(
    function(details) {
        //core logic
        return {
            requestHeaders: details.requestHeaders
        };
    }, {
        urls: ["*://*.foo.com", "https://*/*", oneMoreDomain]
    }, ["blocking", "requestHeaders"]
);
```

RESEARCH METHODOLOGY : Extension Instrumentation

- after identifying the extension, instrument these extensions by overwriting the native webRequest APIs. that helps to capture the modified headers.
- it identifies the background scripts for each extension but not invocations of the hooked event listeners
- To address these issues, a separate analysis is conducted by modifying the extensions to have host permissions for a test domain, which serves all headers considered for analysis.

```
let _onHeadersReceivedListener =
↪ browser.webRequest.onHeadersReceived.addListener;

function _hookResponseListener(callback, filters, opts) {
function _hooked() {

let originalHeaders = arguments;

let newHeaders = callback(arguments);
recordHeaders(originalHeaders, newHeaders); return newHeaders;

}; _onHeadersReceivedListener.apply(this, [_hooked.bind(this),
filters, opts]); }
browser.webRequest.onHeadersReceived.addListener =
_hookResponseListener;
```

RESEARCH METHODOLOGY :URL Scheduling Header Collection

- in this part, instrumented extensions loaded separately in the browser . then it check all hosts to capture the original and modified set of headers at run-time.
- this process avoids server-side randomness. this is one of the fall back of this system.it also not consider obfuscated code in extension.
- To modify HTTP headers it hooks native api.
- For each instrumented extension, the study visits the associated target hosts or Tranco top 100 domains for extensions that operate on all URLs.
- Then the hooked target APIs collect set of the origina headersl and modified headers .Then it relay them to a logging server.
- After every page visit , the cache will be cleared . it gives safeguard to headers to altered.

RESEARCH METHODOLOGY :Detecting Potentially Harmful Extensions

- in this stage , framework retrieves and compare the value of actual and modified header
- within a response , server may sends :
  - different instances of the same header
  - multiple comma-separated values in a single header
  - different instances of the same header
- The framework groups all the distinct values for the given header in a serialized structure . After that it will compares them accordingly.
- For instance, in the case of the X-Frame-Options header, when an extension modifies X-Frame-Options header to enforce a relaxed version, it will replace DENY with ALLOWALL, then the framework can recognize the change and give flag as modifiedHeader
- If there exists no ground truth header after adding an extension in securty header, the analyzer labels the extension as injectedHeader

Results

	2020	2021	2022
Total downloaded extensions	186,434	174,355	180,361
Actual extensions	166,932	154,415	147,334
Extensions with `webRequest`	17,536	10,620	9,298
Extensions with `webRequest` & `webRequestBlocking`	14,821	7,972	6,720
Extensions for dynamic analysis	14,052	7,660	6,505
- targeting <all_urls>	11,824	5,312	4,659
- targeting specific hosts	2,228	2,348	1,846
Extensions with relevant API calls	3,049	3,147	3,145

Table 1: Overview of chrome extensions from each snapshot

CHROME EXTENSION ANALYSIS:Permissions Source Code Analysis

Operational Stage	2020	2021	2022
Considered extensions	14,052	7,660	6,505
Successfully loaded	14,030	7,643	6,486
Registered any handler	9,842	5,172	5,031
Registered relevant handlers	2,735	2,785	2,713
- *onBeforeSendHeaders*	1,695	1,744	1,672
- *onHeadersReceived*	1,639	1,607	1,598
– *both*	599	566	557
Triggered relevant handlers	2,499	2,577	2,553

Table 2: Overview of dynamic extension analysis

- We download three snapshots of chrome and we identify 3,049, 3,147, and 3,145 extensions , which provide some evidence of header interception based on the static analysis.

CHROME EXTENSION ANALYSIS:Result and overall impact

Security Headers	2020 (N = 14,052)				2021 (N = 7,660)				2022 (N = 6,505)				Total
	Inj	Del	Mod	Any	Inj	Del	Mod	Any	Inj	Del	Mod	Any	
content-security-policy	29	189	134	319	23	204	149	339	24	225	157	376	507
content-security-policy-report-only	4	39	24	65	2	47	28	76	1	53	30	83	98
x-frame-options	2	266	13	281	3	300	15	317	4	303	17	321	469
access-control-allow-credentials	7	1	2	10	9	1	4	13	14	1	4	18	22
access-control-allow-headers	14	2	3	16	15	1	5	18	20	1	8	22	29
access-control-allow-methods	24	1	7	26	31	1	10	34	38	1	16	41	46
access-control-allow-origin	57	3	33	66	82	3	38	91	91	4	45	101	121
access-control-expose-headers	3	2	4	6	4	2	4	6	7	2	5	9	11
set-cookie	5	2	4	8	4	2	11	15	8	8	20	28	31
x-content-type-options	1	3	-	5	-	8	1	8	1	9	-	10	13
strict-transport-security	-	7	4	9	-	6	2	6	-	9	2	9	15
referrer-policy	-	1	2	3	-	4	2	6	-	5	2	7	8
permissions-policy	-	1	-	1	-	2	-	2	1	4	1	5	5
cross-origin-opener-policy	-	1	-	1	-	3	1	4	1	4	1	6	6
cross-origin-resource-policy	-	1	-	1	-	2	-	2	2	6	-	8	8
cross-origin-embedder-policy	-	-	-	-	-	1	-	1	1	2	-	3	3

Table 3: Distinct extensions that target different security headers sent by the server-side along with responses.

CHROME EXTENSION ANALYSIS:Result and overall impact

Directive	Injected	Dropped	Modified	Any
script-src	273	274	263	297
object-src	286	290	208	297
frame-src	264	267	262	306
base-uri	284	290	208	293
worker-src	283	289	206	294
connect-src	258	264	247	293
img-src	258	264	233	285
require-trusted-types-for	266	274	207	274
default-src	258	263	239	282
style-src	258	265	226	278

Table 4: Top 10 most frequently impacted CSP directives

CHROME EXTENSION ANALYSIS:Result and overall impact

Security Headers	2020 (N = 14,052)				2021 (N = 7,660)				2022 (N = 6,505)				Total
	Inj	Del	Mod	Any	Inj	Del	Mod	Any	Inj	Del	Mod	Any	
origin	45	2	11	50	67	1	8	70	83	3	11	92	133
referer	105	18	24	124	146	7	32	158	134	10	27	147	240
sec-fetch-dest	-	1	1	2	3	1	8	9	5	1	12	15	16
sec-fetch-mode	-	1	1	2	3	1	4	6	9	1	6	15	16
sec-fetch-site	-	1	1	2	5	1	11	12	12	1	22	32	34
sec-fetch-user	-	1	-	1	2	1	-	2	1	1	-	2	3
upgrade-insecure-requests	-	2	-	2	1	1	-	2	1	-	-	1	3

Table 5: Distinct Chrome extensions that target different security headers sent by client along with requests.

Tested on FIREFOX EXTENSION :that altered security header

Header	Injected	Dropped	Modified	Total
content-security-policy	13	16	29	45
...-report-only	1	2	4	6
x-frame-options	1	19	4	23
ACA-origin	11	0	7	11
ACA-credentials	2	0	1	2
ACA-headers	6	0	5	6
ACA-methods	5	0	3	5
access-control-expose-headers	2	0	2	2
x-content-type-options	0	0	1	1
origin	4	0	2	5
referer	4	6	18	23
sec-fetch-site	0	0	1	1

Table 6: Firefox extensions that modify security headers

Conclusion

- In this paper, we analysis the effect that browser extensions can have on the Web's security by modify- ing the security-related headers.
- there are different mechanism ,like CSP ,XFO, OR COOKIES security attributes which block these extensions when modification occur.
- These automated framework identifies extension that alters security header based dynamical analyzation.
- in this paper , they find 1129 unique extension that alters atleast one security header.from three sets of snapshort of chrome.
- and extented these analysis into Firefox ecosystem and identify 84 unique extension
- still the number of reported extension is small.
- in most-modified headers, CSP and XFO, extensions interfering with those affected many unknowing users.
- And finally author make their pipeline publicly accessible that helps enable stores to issue such warnings automatically