IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 22, 2020, accepted February 18, 2021, date of publication March 4, 2021, date of current version March 26, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3063814

Privacy Model: Detect Privacy Leakage for Chinese Browser Extensions

YUFEI ZHAO[1], LIQUN YANG[1], ZHOUJUN LI[1,3], LONGTAO HE[2], AND YIPENG ZHANG[1]

[1]State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China
[2]National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China
[3]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China

Corresponding authors: Yufei Zhao (zhaoyufei@buaa.edu.cn), Liqun Yang (ylqncepu@163.com), Zhoujun Li (lizj@buaa.edu.cn), and Longtao He (hlt@cert.org.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant U1636211, Grant 61672081, Grant 61370126, and Grant 61906075; in part by the Beijing Advanced Innovation Center for Imaging Technology under Grant BAICIT-2016001; in part by the Fund of the State Key Laboratory of Software Development Environment under Grant SKLSDE-2019ZX-17; in part by the Natural Science Foundation of Guangdong Province, China, under Grant 2019A1515011920; and in part by the Key-Area Research and Development Program of Guangdong Province under Grant 2019B010137003.

ABSTRACT The wide use of browser extensions brings the privacy leakage problem. The previous works detected private data transmission to find privacy leakage in Chrome or Firefox, but the real challenge is to determine whether the transmission is reasonable because the privacy data that existed in transmission does not absolutely mean leaking. To this end, we establish a privacy model for each extension, which contains the sensitive information permitted to be used and servers authorized to communicate with. In order to evaluate the effectiveness of the proposed method, we develop a dynamic browser extension privacy detection framework. It first builds privacy models for extensions and records all network traffic when accessing test pages. Then, the leakage results are presented according to the strict privacy leakage judgment rules. In this paper, the experiments are conducted in a real environment, and our work is verified by 34,095 extensions which are collected from 3 mainstream browsers in China from November 2019 to August 2020. There is a total of 2,983 extensions that exist privacy leakage. We further conduct a comprehensive analysis of the results including calculating the precision, recall, accuracy, and F1 score for each type of leakage, and show the information leaked by different extension categories and the malicious domain name that collecting the users' privacy, as well as the results changing of detection over time.

INDEX TERMS Privacy model, Chinese browser extensions, privacy leakage detection.

I. INTRODUCTION

With the browser extension providing more convenient functions, more and more people begin to use it. The problem of user privacy leakage is becoming increasingly prominent. [9], [23].

The traditional detection method of privacy leakage is to check privacy data in transmission. For some cases, this method is effective, but we can't merely assume that private data transmission is malicious. There is almost no difference between the behavior of benign extension transmitting privacy data for providing necessary services and malicious extension stealing privacy data from the perspective of behavior monitoring.

The associate editor coordinating the review of this manuscript and approving it for publication was Aniello Castiglione.

Whether the transmission is reasonable or not is the most significantly distinguishable. For instance, it is logical that a map extension transmits GPS information, whereas it is abnormal that a game extension collects users' location information. In this paper, two types of privacy data transmission concepts are defined as follows:

Reasonable transmission: An extension probably transfer privacy data to satisfy its function. Users usually grant this type of transmission, or the transmitted privacy data is consistent with the function in this extension.

Abnormal transmission: The transmission is NOT authorized by the users, or, even if authorized by the users, the transmitted privacy data is inconsistent with the function described by the extension, which including the data is transferred to a server unrelated to the extension developer.

The great challenge of the privacy leakage detection method is to judge the rationality of privacy data

transmission, that is to say, whether the transmitted data match the extension function. To solve this problem, this paper proposes the privacy model to describe the conscious range of data. A privacy model contains two parts: data and servers. Data refers to the privacy information allowed to be used, and servers refer to domain names or URLs accepted to connect with. The steps of building a privacy model can be summarized as follows: firstly, we divide extensions into 15 categories according to their functions and classify each extension by its name and description. Secondly, we extract the domain names from the source code and manifest.json file. Finally, we establish a permitted transmission table according to the test pages. By doing this, the scope of each extension transmission can be limited within a reasonable range. If the transmission is within the range, it is a normal extension; otherwise, it is likely to be privacy leakage.

China has a considerable browser market [6], [7], but there are only a few works for extension privacy leakage detection [39]. Although the kernel of Chinese mainstream browsers and that of Chrome are the same, the description of the extensions in the Chinese mainstream browsers and the classification of these extensions according to their functions are quite different due to the differences of languages(most English in Chrome, Chinese in China browser). Therefore, we detect 3 Chinese browsers (QQ browser, 360 secure browser, and 360 speed browser) extensions for privacy leakage. The reason why we do not compare with Chrome is that users cannot directly visit Google in China. In other words, users cannot download extensions from the Chrome application store.

This paper develops a dynamic browser extension privacy leakage detection framework to detect privacy leakage in Chinese browsers. The detection processes can be summarized as follows: firstly, the privacy model is built for each extension; next, the framework installs the extensions on browsers and captures the network traffic when visiting the test pages; finally, the privacy leakage will be estimated according to the judgment rules. To evaluate the effectiveness of our work, we detect extensions collected from November 2019 to August 2020 and display the analysis results in detail.

The main contributions of our paper can be summarized as follows:

(1) We propose a novel privacy leakage detection technique for browser extensions. Using the proposed privacy model, we can judge the rationality of data transmission. The advantages of this technique are: it is more accurate than the traditional methods, and the increased overhead is within an acceptable range.

(2) We develop a dynamic browser extension privacy detection framework based on the Chromium browsers and provide a detailed technical description of it, which can fully launch the entire detection process and demonstrate the authenticity of the leak.

(3) The effectiveness of the proposed framework is verified by large-scale testing in the real-world environment. We conduct a 10-month detection for QQ browser, 360 secure browser, and 360 speed browser, where 34,095 extensions are tested, and we find 2,983 privacy leakage instances. Additionally, we calculate the precision, recall, accuracy, and F1 score for each leakage type and show the direction of the leak and the malicious domain names, as well as the results changing of detection over time.

II. BACKGROUND

We introduce the permissions in manifest.json by reviewing the structure of an extension file. The extensions have exactly the same structures on the chromium-kernel-based browsers, and all the files are packed in a CRX file, which includes all.js,.json,.css,.html, and other files. The source code of an extension existing in the script folder, and every extension has a JSON-formatted manifest file, named manifest.json that provides essential information [16], including metadata such as the name and version, and specify aspects of functionality (e.g., background scripts, content scripts, and browser actions).

There are multiple fields for extension permissions in the manifest.json, and we will describe them separately.

```
{
\ldots \ldots
"content\_scripts": [{
"matches": ["http://*.*","https://*.*"],
"js": "go.js",
"run_at": "document_start"}]
"background": {
"scripts": ["scripts/background.js"]}
"permissions": [
"http://*.*",
"https://*.*",
"webRequest"]
"content_security_policy":
"script-src 'self'
http://www.foo.com 'unsafe-eval';"
\ldots \ldots
}
```

Content Scripts: By specifying a list of content_scripts, the extension runs different JavaScript files in the context of the page that satisfies the matching criteria [12]. In the above, the sentence in"content_scripts" means the go.js will be executed when the browser accesses any page.

Background: A background page is loaded when it is needed and unloaded when it is idle. It is typically driven by events (such as navigating to a new page, removing a bookmark, or closing a tab) and then reacts to the specified instructions [15]. In the above, an extension may listen to all network requests via background.js and record the requested information. Background Page is usually transparent to the users and typically contains the logic and state of the browser session.

Permissions: To use most chrome.* APIs, an extension must declare its intent in the "permissions" field of the manifest.json file. Each permission can be either one of a list of known strings (e.g., geolocation) or a match pattern that

gives access to one or more hosts [14]. For example, ''permissions'' in the above, the webRequest permission enables the extension to ''observe and analyze traffic and to intercept, block, or modify requests in-flight'' [11]. The URLs that the extension requests to access also need to be declared in the permissions field, namely host permission. Also, many extensions set the host like ''*http://*/**'' or ''*https://*/**''. These hosts allow the extension to access any URL indiscriminately.

Content Security Policy (CSP). In order to mitigate a large class of potential cross-site scripting issues, ***CSP*** acts as a white list to allow resources to be loaded or executed by the website. ***CSP*** can also specify other options, such as whether to allow the page to execute eval or embed inline JavaScript [13]. For instance, in the above, the extension can execute the eval function on the www.foo.com page.

III. DATA COLLECTION AND ANALYSIS

In the previous work [39], it has been found that the extensions divulge the users' privacy. However, due to Baidu browser stopping service and the UC browser users are few, we continue to analyze the 360 browser which accounts for more users, and collect and sort out 16,547 extensions of three (QQ browser, 360 secure browser, and 360 speed browser) for six months, from May 2019 to October 2019.

A. EXTENSION CLASSIFICATION

In reality, extensions are coarse-grained classified according to the functionality in the three browsers. But after analyzing manually, we believe that the categories in application stores are inaccurate. We collect the name, short-description, and detailed-description of each Chinese extension and discover that many extensions are similar.

The name and description of a Chinese extension can illustrate its functionality and usually have unique keywords. Since there is no clear boundary in Chinese, we need to complete Chinese natural language processing(NLP) through Chinese word segmentation. The tool jieba [18] is used to perform Chinese word segmentation.

Firstly, the extensions are labeled by manual analysis, and different keywords are determined for each category. Next, TF-IDF (Term Frequency-Inverse Document Frequency) value is calculated for every key word, and the sum of key words in one category is set to be a threshold. The monthly collection of the extension data from three browsers is seen as a round of experiments to adjust the keyword range and threshold value for each class. After a half-year experiment, the classification results can meet the needs of subsequent testing.

For instance, key words are extracted from an extension as $K_i, i = 1, 2, 3 \ldots n$. Next, the TF-IDF algorithm is adopted to calculate the TF-IDF value for each key word.

$$\begin{aligned} TF - IDF(K_i) &= TF(K_i) \times IDF(K_i) \\ &= \frac{O(K_i)}{T} \times log(\frac{N}{N(K_i)}) \end{aligned} \tag{1}$$

$$sum = \sum_{i=1}^{n} TF - IDF(K_i) \tag{2}$$

$O(K_i)$ is the occurrence number of K_i from extracted words for an extension; T is the total number of extracted words for an extension; N means the total number of documents in the corpus, namely the number of analyzed extensions; and $N(K_i)$ is the number of extensions which extracted from the words containing K_i. Then the TF-IDF values of each keyword are summed. If the sum is greater than the threshold, the extension is classified into this class.

According to the above calculation results, extensions can be classified into 15 classes. Each class has unique functionality that are different from others. For example, the function of the *Open* is to help users rapidly open web pages or programs. This kind of extension, in principle, does not need any privacy data. The *Shopping assistant*, however, can compare the prices of the same product from different platforms(such as Taobao, Jingdong, and so on), so it needs to collect the history of a user's access to different shopping sites. In practice, an extension may belong to more than one class but not more than three classes.

B. NETWORK MONITOR

In addition to running local scripts, an extension can also download and execute the content from the website. Monitoring the network activity of the extension is critical for a complete analysis because the URLs retrieved can be computed at run-time.

Almost all extensions use the HTTP protocol to communicate with their servers, and a few may use the HTTPS protocol. Many related works adopt transparent proxy to capture plaintext [19], [33]. However, only a fraction of traffic can successfully be obtained through the man-in-the-middle(MITM) attack. The MITM fails to get the plaintext for some strict protection sites taking HSTS (HTTP Strict Transport Security). To solve this problem, instead of adopting transparent proxy to obtain the plaintext information of HTTPS, the source code of Chromium browser is instrumented to gather complete network behaviors when visiting web pages, especially the data sent by the browser and the domain name of the server connected. Fig. 1 is the pseudo-code of modification of the HTTP parse function.

C. TYPES OF PRIVACY DATA

We check network traffic for the 16,547 extensions(from May 2019 to October 2019) through an instrumented browser by visiting 5 web pages (baidu.com, taobao.com, jd.com, weibo.cn, and icbc.com.cn), which includes a search engine, two shopping sites, a social platform, and a bank website. The results show there still exists various degrees of private leakage in the extensions. Malicious extensions do not directly steal important privacy information(e.g., username, password, or credit card number) but record the accessing history to portray users or collect browser or system information such as screen resolution, operating system type, etc.

According to the observed results, the privacy data are divided into five categories: browsing history, geographic location information, browser information, operating system

TABLE 1. Extension classification.

Num	Class	Description	Private Data
1	Protection	Private protector; malicious detection	NULL
2	Block	AD block; Sites block; URL block	NULL
3	Geographical Location	Weather Forecast; Map; Air Quality	Geographic location information; Network address information
4	Open	Open the web pages or programs	NULL
5	Shopping assistant	Price comparison; Identify fakes	Browsing history (of shopping websites)
6	Language and translation	Translation; Dictionary; Grammar check; Language learning	Geographic location information (language)
7	Enhancement	Web page function enhancement; System expansion: Tab theme, Page reading, Right button extend; Web page forensics; Page screenshot	NULL
8	Social Contact	Wechat;Weibo	Operating System Information; Geographic Location Information
9	Develop Tool	Programming development or assistant tools	NULL
10	Fun	Game; Sports; Video; Music; Picture; E-book	NULL
11	Proxy	Proxy switch; VPN; Break the great wall	NULL
12	Crawler	Web page Crawler	Browsing history
13	Web page history	Record user's history	Browsing history
14	Password management	Password manager and remember	Browsing history;User Login Information
15	IP/Domain/MAC address	Provide a network address	Network address information

http_stream_parser.cc

```
……
//Check body part of request
if HasRequestBody(Request) then
    data ← GetHeader(Request)
    ……
    //Extract body from request and append it to data
    data ← data + ParseAndGetBody(Request)
else
    data ← GetHeader(Request)
//Log TimeStamp and data in log file
Log(TimeStamp, data)
……
```

FIGURE 1. The modification of the HTTP parse function.

information, and Network address information. The corresponding data leakages are called history leakage(HL), location leakage(LL), browser leakage(BL), OS leakage(OL), and network address leakage(NL).

Browsing history refers to the history of user's access to web pages; geographic location information includes the latitude and longitude or base station information, and the user's language; browser information includes the browser type and version; operating system information consists of operating system type, version, screen resolution, etc., and the network address information contains the IPv4 address or MAC address of the device. The privacy data allowed to be transferred by each class is shown in Table 1.

IV. DETECTION

In this section, we introduce the detection method and establish a dynamic browser extension privacy detection framework. Fig. 2 shows that the procedures of the detection method in which there are 5 parts: classification, preprocessing, dynamic detection, traffic analyze and leakage judge. Classification can determine the categories and functions of the extensions, and the privacy model is built through preprocessing. Test pages will be visited randomly in dynamic detection, and privacy data is carefully searched from traffic.

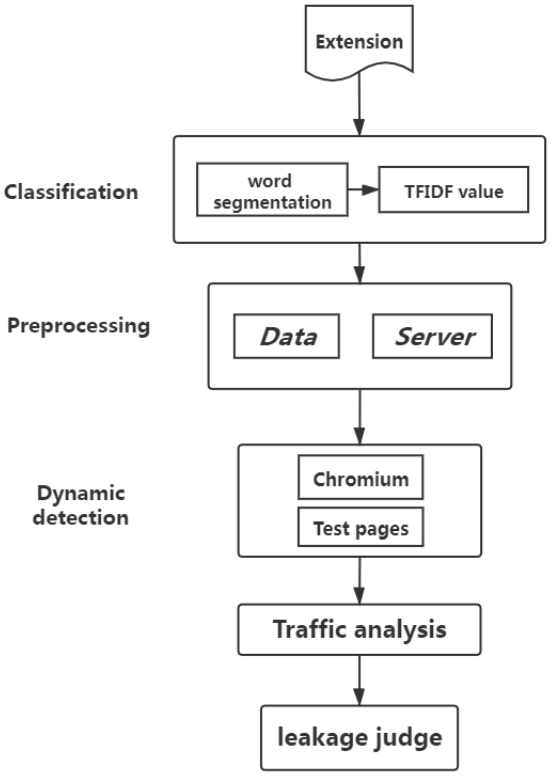

FIGURE 2. Procedures of detection method.

Finally, the leakage detection rules are estimated to determine whether the leakage exists. Fig. 3 depicts the architecture of it, which can analyze an extension according to the following steps. First, extensions are pre-processed to build privacy models and determine the test pages for evaluation. Then instrumented browser records the network traffic while accessing the test pages after installing the extension. Finally, the extension is labeled as normal or malicious by analyzing whether the behavior matches the privacy model.

A. EXTENSION PRE-PROCESSING

We statically analyze the extension to establish the privacy model and determine the test pages to be visited during detection.

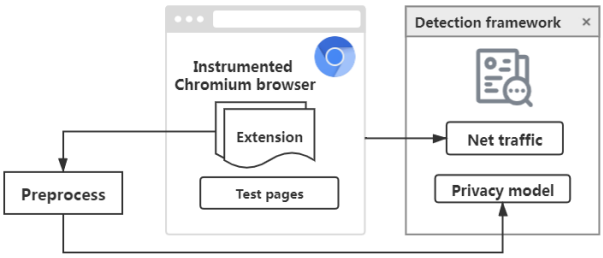

FIGURE 3. The architecture of the dynamic browser extension privacy detection framework.

TABLE 2. Contents of privacy data.

Privacy Data	Content
Browsing History	baidu.com, taobao.com, jd.com,112.74.12.*, etc.
Geographic Location Information	zh-cn, locale, lang, country, etc.
Browser Information	chrome,chromium,60.0.3094.0, 73.0.3683.103,currentTag,ch60,ch73,etc.
Operating System Information	win,Windows,Win64,screenwidth,etc.
Network address information	211.94.162.*, 207.148.75.*,etc.

1) BUILDING PRIVACY MODEL

The privacy model includes two parts: *data* and *server*. *Data* refers to the type of private data allowed to be used, and *server* refers to the server to which the extension is allowed to submit data.

We perform word segmentation for the name, short-description, and detailed-description of each Chinese extension, and use the algorithm in 3.A to complete the classification. As mentioned in Section 3.C, different classes have different types of privacy data. This confirms the *data* that each extension can apply. When checking the network traffic from May 2019 to October 2019, we find that the privacy data content has a certain regularity, not as much as expected. Therefore, we summarize the contents of all types of private data are shown in Table 2.

For example, an extension belongs to the shopping assistant category, and it can record browsing history information only for shopping websites, on account of it needs to transmit shopping history to implement the price comparison. So, it is reasonable to upload taobao.com (shopping website in China) URL, but it is unusual to record user's access to baidu.com(search engine in China).

As for *server*, we extract the server domain name from manifest.json. Most extensions' manifest.json file has a homepage_url field, from where we can extract the primary domain name. If a valid server cannot be extracted from the manifest.json file, no data should be transferred to any server.

2) TEST PAGES

The stealing behavior of extensions is very stealthy and only occurs when users are visiting specific pages. In order to trigger theft behavior as much as possible, we carefully select some websites that are frequently used in our lives as test pages and specially constructed a login page. The number of test pages is an issue worth discussing. In prior work, each extension used dozens to hundreds of test pages, but from the perspective of simulating user behavior, it is difficult for a user to access a large number of pages in a short time. Therefore, in the test process of this article, each extension only accesses 10 Pages, including shopping websites (taobao.com, jd.com), search engines (baidu.com, so.com), banking (icbc.com.cn), social (weibo.com), entertainment (iqiyi.com), education (tsinghua.edu.cn), government (www.gov.cn) websites, and a login page (112.74.12.*). All test pages' URLs belong to *Data* in privacy model. During the detection, each extension will randomly visit these 10 web pages and save the network traffic during the visit.

B. TRAFFIC ANALYSIS

The plaintext traffic can be obtained through the instrumented browser, but it does not mean that the private data can be found directly in the plaintext. According to our manual analysis on the known malicious extensions, the most commonly used encoding techniques in Chinese browsers are base64 and twice base64.

One problem that needs to be solved is how to identify different combinations of the known encodings and automatically decode them into textually meaningful strings (mainly for URL and base64 decode). Therefore, we extract parameters from the HTTP request and continuously try to decode them until the program exits with an error or reaches the required number of iterations. From the preliminary analyses, when the number of iterations reaches 20, the requirements for the subsequent discovery of privacy data leakage have been met.

In fact, stealers can bypass our detection scheme through customized encryption or multiple encodings. Many existing papers have faced similar problems and have the same limitations as our work, but they are also providing critical practical values for the users.

C. LEAKAGE JUDGEMENT

We compare the data and server extracted from the traffic with the privacy model (PM) and judge whether there is a privacy leak according to the following rules.

$$\begin{cases} (a) \quad data \notin PM, or, server \notin PM \rightarrow malicious \\ (b) \quad data \in PM, and, server \in PM \rightarrow normal \end{cases}$$

In theory, the final judgment needs to be related to the extension privacy policy. If there is a detailed description of the collected data and usage in the privacy policy, then the corresponding data needs to be added to the privacy model. However, in the three browsers we tested, each extension does not have a separate privacy policy. We strongly recommend that the browser application store requires the extension developer to provide a detailed privacy policy description.

V. RESULTS AND ANALYSIS

In order to test a wide range of Chinese browser extensions and evaluate the effectiveness of the proposed dynamic detection framework, 34,095 extensions are detected from November 2019 to August 2020. In particular, the update frequency

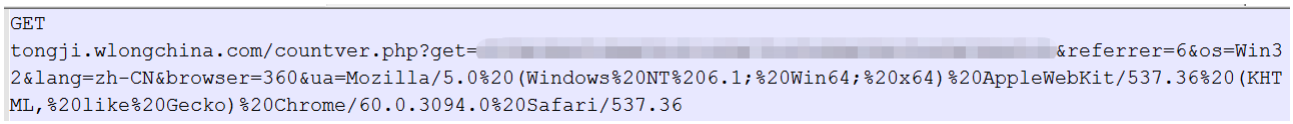

FIGURE 4. Upload user-agent in URL.

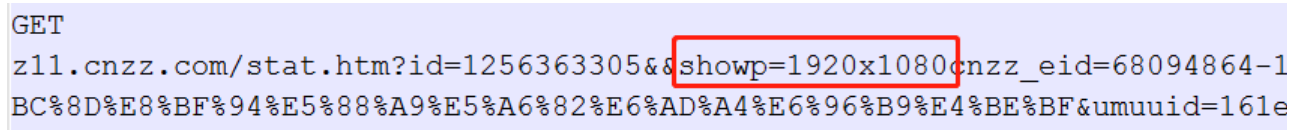

FIGURE 5. Upload screen resolution in URL.

is low for application stores of QQ browser, 360 secure browser, and 360 speed browser. If a comprehensive test is performed every day, a large number of repeated results will be produced. Therefore, a comprehensive inspection is performed every 30 days.

A. LEAKAGE RESULTS

During the 10-month test period, there are 2,983 privacy leak extensions detected. The results are shown in Table 3. Particularly, the sum of the five types of privacy data leakage will be greater than the number of leakage extensions because some extensions have multiple types of the leak at the same time. Additionally, we statistic the true negative (TN), true positives (TP), false negative (FN), and false positive (FP) of each type and calculate the precision, recall, F1 score, and accuracy of detecting all privacy data leakage categories with the following formulas.

$$P(precision) = TP/(TP + FP)$$
$$R(recall) = TP/(TP + FN)$$
$$F1 = 2 * P * R/(P + R)$$
$$accuracy = (TP + TN)/(TP + TN + FP + FN)$$

As for history leakage, some extensions record user's browsing behaviors, including the domain name, URLs, and even the order of visited pages, and usually upload the browsing history to unrelated third parties. The examples of HL will be detailed explained in the *leakage pattern*. We also expect to find extensions that leak geographic location information, but none of the extensions transmit any latitude and longitude information. However, many extensions carry language information in the transmitted data. In theory, the approximate location can also be located through the language used, so this situation is also regarded as a location leak. The results show that many extensions will send user language information, which will also cause geographical location leakage, and LL is the largest number of all private data leakage.

Browser information and operating system information often appear together in network traffic. In fact, the User-Agent field of the HTTP protocol contains this part of the information and should not be regarded as private information under normal circumstances. However, in the detection, we found that the information collected by the malicious extension far exceeds the scope of User-Agent, including but not limited to screen width, resolution, color depth, and detailed version information. For example, in Fig. 4, an extension (VIP watch video in 360 secure browser) uploads User-Agent in URL to the unrelated server, and in Fig. 5, the screen resolution is uploaded(7Rebate in 360 speed browser).

TABLE 3. Detection results.

	QQ browser	360 secure browser	360 speed browser	total number
number	5495	15296	13304	34095
total leakage	518	1307	1158	2983
HL	110	446	381	937
HL-TN	5385	14819	12905	
HL-TP	110	446	381	
HL-FN	0	31	18	
HL-FP	0	0	0	
HL-precision	100.00%	100.00%	100.00%	
HL-recall	100.00%	93.50%	95.49%	
HL-F1	100.00%	96.64%	97.69%	
HL-accuracy	100.00%	99.80%	99.86%	
LL	188	746	550	1484
LL-TN	5307	14550	12754	
LL-TP	188	746	550	
LL-FN	0	0	0	
LL-FP	0	0	0	
LL-precision	100.00%	100.00%	100.00%	
LL-recall	100.00%	100.00%	100.00%	
LL-F1	100.00%	100.00%	100.00%	
LL-accuracy	100.00%	100.00%	100.00%	
BL	271	471	389	1131
BL-TN	5202	14809	12907	
BL-TP	241	444	370	
BL-FN	22	16	8	
BL-FP	30	27	19	
BL-precision	88.93%	94.27%	95.12%	
BL-recall	91.63%	96.52%	97.88%	
BL-F1	90.26%	95.38%	96.48%	
BL-accuracy	99.05%	99.72%	99.80%	
OL	229	322	214	765
OL-TN	5266	14974	13090	
OL-TP	229	322	214	
OL-FN	0	0	0	
OL-FP	0	0	0	
OL-precision	100.00%	100.00%	100.00%	
OL-recall	100.00%	100.00%	100.00%	
OL-F1	100.00%	100.00%	100.00%	
OL-accuracy	100.00%	100.00%	100.00%	
NL	53	72	58	183
NL-TN	5442	15224	13246	
NL-TP	53	72	58	
NL-FN	0	0	0	
NL-FP	0	0	0	
NL-precision	100.00%	100.00%	100.00%	
NL-recall	100.00%	100.00%	100.00%	
NL-F1	100.00%	100.00%	100.00%	
NL-accuracy	100.00%	100.00%	100.00%	

```
GET
r1---sn-2x3een7z.gvt1.com/edgedl/release2/chrome_componer
6358045.data.crx2?cms_redirect=yes&ip=171.120.24[redacted]&ipbi
m&pl=18&shardbypass=yes
```

FIGURE 6. Upload IP address in URL.

TABLE 4. Leakage percentage in all categories.

category	number	leakage percentage
Protection	266	2.63%
Block	715	9.51%
Geographical Location	364	1.65%
Open	310	5.48%
Shopping assistant	4203	26.91%
Language and translation	946	19.77%
Enhancement	21365	3.68%
Social contact	153	22.88%
Develop tool	2143	12.88%
Fun	2592	10.11%
Proxy	281	16.01%
Crawler	147	27.21%
Web page history	98	38.78%
Password management	453	16.34%
IP/Domain/MAC address	59	18.64%
TOTAL	34095	

We detect the IP address and MAC address of the test machine from the traffic. Surprisingly, extensions that exceed our expected number will directly upload the user's IP address, which is a severe leak without clearly telling the user. An instance is shown in Fig. 6. The IP address is sent to the domain name *gvt1.com* the most times. Although these 3 browser extensions are mainly designed for Chinese users, the IP addresses of some of them are still uploaded to Google servers.

We make statistics on the proportion of leaked extensions in each category in Table 4. By comparing the leakage percentages in the extension category, it is clear that certain types of extensions are more prone to privacy data leakage. The top three categories that account for the percentage of leaks are: web page history, crawler, and shopping assistants. Sending users' browsing history, browser information, and operating system information to third-party servers is the main reason for their privacy data leakage. Our analysis also reveals a worrying volume of disclosure, and almost all extension classes have leakage behavior. Fig. 7 shows the leakage of each extension type, Enhancement, Shopping assistant, and Fun classes contribute the most privacy breach cases. In fact, the number of Enhancement extensions accounts for 62.66% of the total detection volume, although the leakage percentage is low, the absolute number is the largest. Shopping assistant, and Fun are also the hardest-hit areas for privacy leaks, and they tend to be history leakage, browser leakage, and OS leakage. The attendance of Develop tool is surprising, especially the detection discoveries that it leaks more network address information, and we recommend that developers make use of development tools locally. Although the number of leaks of other types of extensions is relatively small, the Protection and Block categories should not appear in the test results. This also reminds users again that they must pay more attention to the privacy and confidentiality of extensions installed from unofficial channels.

Table 5 presents the top 3 servers, which collect privacy information, for every category of leakage type. The history leakage and the browser leakage servers are the same because these two kinds of information usually be transmitted together, which details are in section v-B. The domain names *starwebnet.com, datarating.com, and webovernet.com* are all malicious servers, and they first assign a unique subdomain to each extension and then gather data through twice Base64 encoding. The domain names such as *gvt1.com, google-analytics.com, and doubleclick.net* are registered and used by Google Inc. Even though QQ and 360 browser extensions are mainly designed for Chinese users, some privacy (language, operating system, and network address) are still uploaded to Google servers. Not all servers collect private data maliciously. It is possible that some extensions call some "contaminated" public underlying libraries to cause upload behavior. In this instance, we suspect that most of these extensions replicate the source code from the Chrome app store, and as a result, some extensions exhibit the same behavior as Chrome extensions to upload information to Google servers. Domain name *cnzz.com* is from the world's largest Chinese Internet data statistical analysis service provider. Similarly, *mixpanel.com* is a data tracking and analysis company that allows developers to track various user behaviors. An interesting phenomenon is that extensions upload user IP addresses to *youdao.com*, a Chinese intelligent learning company dedicated to providing user-oriented learning products and services. We are uncertain for the cause of this occurrence, and it may be that the IP address roughly determines the location of the user and the type of language used.

In addition, the detection results about five types of privacy data leakage need further explanation. There are some false negative results in history leakage because some extensions make use of customized encryption in transmission, the detection framework cannot recognize encrypted information from network traffic. There are also no false negative or false positive in location leakage, OS leakage, and network address leakage. The reason is that these types of privacy data are specific and explicit (such as the user's IP address), and the data is hardly encrypted, so it is easy to identify from network traffic. The result of browser leakage contains both false positives and false negatives, and this is due to the similarity between browser information and extension information. For example, it is difficult for the framework to determine whether this parameter represents a browser type or an extension type like $f = chrome$.

B. LEAKAGE PATTERN

We have concluded three commonly used private data transmission patterns through careful analysis of network traffic, namely plaintext transmission, encoded transmission, and customized encrypted transmission.

Plaintext transmission refers to the transmission of data through the GET or POST method of the HTTP protocol. For example, the detection framework will randomly access the test page, and the malicious extension will transmit the visited

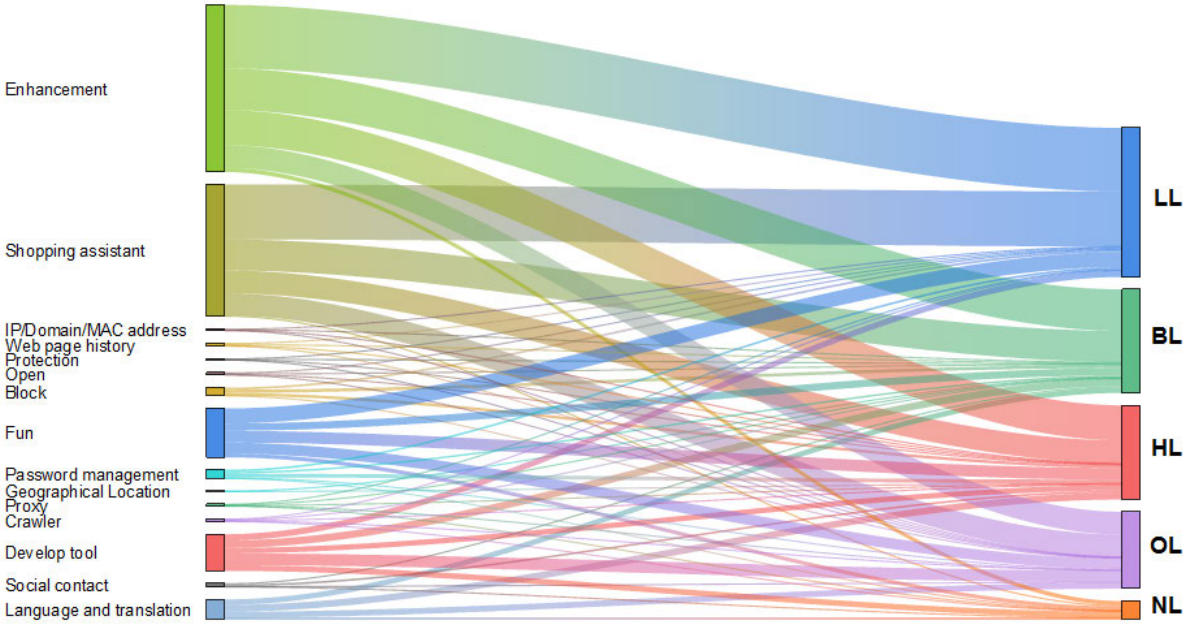

FIGURE 7. Leakage of each extension type.

TABLE 5. Top3 servers for all leakage categories.

Leakage type	Servers
HL	starwebnet.com datarating.com webovernet.com
LL	cnzz.com google-analytics.com mixpanel.com
BL	starwebnet.com datarating.com webovernet.com
OL	cnzz.com mixpanel.com doubleclick.net
NL	gvt1.com cnzz.com youdao.com

URL to the server in plaintext, in order to record the user's browsing history. As shown in Fig. 8 and Fig. 9. The plaintext mode is poorly concealed and easily to be detected or identified. Therefore, some malicious extensions use encoding to transmit private data, which increases the difficulty of detection. The most commonly used encoding method is Base64 encoding, and we also found twice Base64 encoding in the traffic. In Fig. 10, the privacy data is encoded in standard Base64, After decoding by Base64, the plaintext is:"pid":"1520927358520vLbKy54Q6o","title":"Taobao", "url":"https://www.taobao.com/","refer":"", "timeIn":1520927358520. The extension not only uploads the URL of the webpage, but also uploads the titles of

```
POST /ydy/1.0.5.1 HTTP/1.1
Host: www.meinvgong.com:9090
Connection: keep-alive
Content-Length: 28
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64;
Origin: chrome-extension://jgjokgppkphjkpglhhfg
content-type: application/x-www-form-urlencoded
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8

location=https://www.jd.com/
```

FIGURE 9. Transfer browsing history through POST method.

the pages. In Fig. 11, we can see that the extension has strengthened the confusion of the privacy data. After two consecutive base64 decodings, we can see that the parameters *q* and *prev*. *q* represents the URL of the currently visited page, and *prev* represents the URL of the previously visited page. It is not difficult to infer from the figure that the first page we visited is baidu.com, followed by taobao.com. In other words, with the extension enabled, user browsing history and accessing order can be fully recorded.

The third leakage pattern is customized encrypted transmission, which is more concealed and more difficult to detect. In the check period, we found that some extensions adopt XOR encrypted transmission, which can bypass the current detection framework, which is also one of the reasons for

```
GET /plugin.php?version=1.0&type=flaginfo&hostname=www.taobao.com&:
Host: www.utlsapi.com
```

FIGURE 8. Transfer browsing history through GET method.

```
GET
/log3.gif?i=JTdCJTIycGlkJTIyJTNBJTIyMTUyMDkyNzM1ODUyMHZMYkt5NTRRNm8lMjIlMkMlMjJ0aXRsZSUyMiUzQSUyMiVFNiVCNyU5OCVFNSVBRSU5RCVFNyVCRCU5MSUyMC01MjAlRTYlQjclOTglRUYlQkMlODElRTYlODglOTElRTUlOTYlOUM1RTYlQUM1QTIlMjIlMkMlMjJ1cmwlMjIlM0ElMjJodHRwcyUzQSUyRiUyRnd3dy50YW9iYW8uY29tJTJGJTIyJTJDJTIycmVmZXIlMjIlM0ElMjIlMkMlMjJ0aWllSW4lMjIlM0ExNTIwOTI3MzU4NTIwJTdE&u=dSUzQTFjZjUxMWRlODY5ZGQ3ZjQwYmJlZjAyNzIwZWJlZTMzJTJDMTUyMDkyNzM1OTI2MSUzQm51JTNBJTNCaSUzQQ== HTTP/1.1
Host: 519914-0.sh.1252135381.clb.myqcloud.com
Connection: keep-alive
```

FIGURE 10. Privacy data in base64 encoding.

```
POST s661.starwebnet.com/related
row_body:e=Y3owMk5qRW1iV1E5TWpFbWNHbGtQV1EzZERsM04wTmphR0V4ZEZsbGJTWnpaWE56UFRRME16SXpPRGs1TmpRM01qY3dORFF
3TUNaeFBXaDBkSEJ6SlROQkpUSkdKVEpHZDNkM0xtSmhhV1IxTG1OdmJTVXlSaVp3Y21WMlBTWnNhVzVyUFRBbWMzVmlQV05vY205dFpTW
m9jbVZtWlhKbGNqMG1kRzEyUFRNd01UYz0=

debase64:s=661&md=21&pid=d7t9w7Ccha1tYem&sess=443238996472704400&q=https://www.baidu.com/&prev=&link=0&sub
=chrome&hreferer=&tmv=3017

POST s661.starwebnet.com/related
row_body:e=Y3owMk5qRW1iV1E5TWpFbWNHbGtQV1EzZERsM04wTmphR0V4ZEZsbGJTWnpaWE56UFRRME16SXpPRGs1TmpRM01qY3dORFF
3TUNaeFBXaDBkSEJ6SlROQkpUSkdKVEpHZDNkM0xuUmhiMkpoYnk1amIyMGxNa1ltY0hKbGRqMW9kSFJ3Y3lVelFTVXlSaVV5Um5kM2R5N
WlZV2xrZFM1amIyMGxNa1ltYkdsdWF6MHdKbk4xWWoxamFISnZiV1VtYUhKbFptVnlaWEk5Sm5SdGRqMHpNREUz

debase64:s=661&md=21&pid=d7t9w7Ccha1tYem&sess=443238996472704400&q=https://www.taobao.com/&prev=https://ww
w.baidu.com/&link=0&sub=chrome&hreferer=&tmv=3017
```

FIGURE 11. Privacy data in twice base64 encoding.

```
POST /shops/buy_helper_report? HTTP/1.1
Host: bj.gj.qq.com


d=OGE3IiRheWEhKW0gKzEsLiZtJyY3IiovbTcuIi8vbS
Lm1yc3Nze3Z7bXJzc3N0cXZtcXJtJjMIdHQnZSonfnZx
NwdXBwc2UqMBwhfnJlMS1%2BciV2e3QhIHN3InUne3Yi
```

FIGURE 12. Custom encrypted transmission.

false negative. As shown in Fig. 12, the data looks to be base64 encoded, but it is still garbled after decoding. Manually analyze the extension's source code and find that the private data is first XORed with 0 × 43 byte by byte and transmitted in base64 encoding. The decrypted plaintext is: {"tag":"bj.chrome.detail.tmall.com.item.js.loading", "itemurl":"https://detail.tmall.com/item.htm?spm=a220 m.1000858.1000725.21.epK77d&id=521803268106 &skuId=3108311119308&areaId=440300 &user_id=2260280385&cat_id=50036330&is_b=1 &rn=1f587bc04a6d85a6554cd92702b9ae15", "uid":"152602536518088539244131123366779"}.

We found the URL of tmall.com, but this URL is not on the test pages because the extension will automatically open this page when it is enabled. The following manual analysis also found that when accessing the homepage of a common shopping website, the URL will not be uploaded. Only when the specific product page is accessed, the browsing record is uploaded to the server.

C. LEAKAGE OVERTIME

In addition, we analyzed the changes of 3 browser extensions over time and the trend of privacy leak rates over time to understand whether the measured privacy leaks are accidental or have become a stable attribute of the browser extension ecosystem. We have calculated the number of monthly

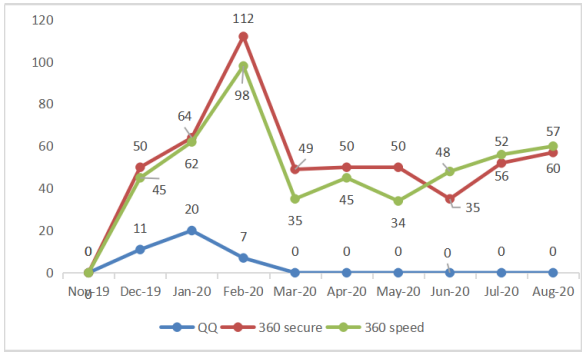

FIGURE 13. Updated statistical results in 3 browsers.

expansion updates, as shown in Fig. 13. It is not difficult to find that these 3 browser extensions are rarely updated, with an average of only 2%-3% extension updates every month. Significantly, the number of updates in January and February is vast, and the QQ browser has hardly been updated in other months, 360 secure browser and 360 speed browser have an average of 50-60 extension updates every month. The reason for a large number of changes in January and February is the combined impact of the Chinese Spring Festival and COVID-19, which has caused many people to stay at home. Developers have more time to update extensions and develop new extensions to illustrate the spread situation of COVID-19.

The lower update frequency results in barely changing in the leakage situation. Once the privacy leakage behavior of an extension is detected, it is possible that the extension will continue to leak behavior during the detection period. For these very stable browser extension ecosystems, our detection framework is very effective and can quickly detect privacy leaks for a long time.

VI. RELATED WORK

The browser extension security analyses can be divided into two categories: the static analysis and the other is the dynamic analysis. Previous works about static analysis are shown as follows. Guha *et al.* [10] formalized the semantics of policies in terms of a safety property on the execution of extensions and developed a verification methodology. Dev and Jevitha [8] used the STRIDE approach, developed by Microsoft, to identify each Chrome specific API's possible threats.

Static analysis methods are easily interfered with and result in low detection accuracy. Therefore, researchers used dynamic analysis to run extensions in a monitored environment and record all behaviors of extensions. HULK [19] first analyzed the categories of malicious extensions and used event-driven analysis to complete a large number of analysis tasks in a short period of time. Starov and Nikiforakis [33] further defined the information leakage behavior and classified the privacy information leakage into four categories: historical record leakage, search query leakage, form data leakage, and other extended application information leakage, 9839 Chrome extension apps were analyzed through dynamic testing and operational simulation. Weissbacher *et al.* [36] designed a machine learning system based on the topic of historical information leakage and detected the history stealing behavior in an unsupervised learning manner, and conducted in-depth research on the behavior of the tracker through the set honeypot domain address.

In order to make the analysis more precise, researchers have introduced data flow analysis methods. Chang and Chen [4] adopted dynamic taint to track the sensitive information leakage at runtime. Chen and Kapravelos [5] enhanced traditional taint analysis using information gathered from static data flow and control-flow analysis of the JavaScript source code.

In the classification of malicious behavior of extensions, Zhao *et al.* [38] analyzed the user information stealing behavior of 28 extended applications in detail. Jagpal *et al.* [17] then explained Google's detection strategy for the Chrome browser extensions and troubleshooting results. Barth *et al.* [1] investigated the API of the extension, divided the behavior of the malicious extensions in five levels, and matched the behavior of the extension with the API according to the generated security risks; Pardo *et al.* [25] expected to establish a policy to implement monitoring during the running of the extended application. Rizothanasis *et al.* [29] judged user's behavior by modifying the browser source code. Nikiforakis *et al.* [22] analyzed the code of three popular browser-fingerprinting code providers and revealed techniques that allow websites to track users without client-side identifiers. Xing *et al.* [37] developed Expector, a system that automatically inspects and identifies browser extensions that inject ads, and then classifies these ads as malicious or benign based on their landing pages. And many works aimed at Firefox browser [2], [24].

Apart from the privacy leakage of extensions, prior works also explored finger-printability dimensions or attacks to extensions or browsers [21], [27]. Starov and Nikiforakis [34] presented XHOUND to investigate and quantify the finger-printability of Chrome extensions through DOM-based modifications. Karami *et al.* [20] extended the approach by automatic generating behavioral fingerprints. Sjösten *et al.* [32] studied a class of extension revelation attacks, where extensions reveal themselves by injecting their code on web pages, and invalidated the randomized ID of Firefox, and the attacker can use the random ID as a reliable fingerprint of the users. Roesner *et al.* [30] conducted a full study of the web track for the first time and developed a client-side method for detecting and classifying five kinds of third-party trackers. Rauti implemented browser extensions for Mozilla Firefox to demonstrate man-in-the-browser attacks against Ajax applications [28]. Varshney *et al.* [35] studied spyware and fraud for extensions, and developed a lightweight malicious extension detection system using the collected malicious behavior characteristics, and statically detected extensions through feature data sets. Sanchezrola *et al.* [31] presented a timing side-channel attack against the access control settings and an attack that takes advantage of poor programming practice, affecting a large number of Safari extensions. Perrotta presented a botnet framework based on malicious browser extensions and provide attacks that can be launched in this framework [26]. Castiglione *et al.* [3] analyzed and designed a steganographic system to create a covert channel to block browser fingerprinting tracking.

VII. CONCLUSION AND DISCUSSION

This paper concerns the problem of privacy leakage induced by Chinese browser extensions. The traditional detection method is to check whether there are privacy data in the transmission, in this paper, we present privacy model that clarifies the usage scope and transmission range of private data to estimate the reasonability of transmission. The privacy model is a heuristic detection method. By analyzing the extension data in the first five months, we find the privacy data leakage characteristics and detect the extensions in the following 10 months. Firstly, through manual analysis in the early stage, the whole extensions are divided into 15 classes, and the privacy data are classified into 5 categories. And then, develop a dynamic detection framework and verify its effectiveness by performing a consistently large-scale study on mainstream browser extensions in China from November 2019 to August 2020. As a result, 2,983 malicious extensions are found. We also illustrate the test results in detail. The results show that the update frequency of these three browsers is very slow, and our detection method has nice applicability.

Although the current framework detection results are satisfactory, the generalization of the detection algorithm needs to be further verified. Due to the low update frequency of QQ browser, 360 safe browser, and 360 speed browser,

many malicious extensions may exist long-term privacy leaks, that is the main reason for the significant detection results. There is a potential danger that once the frequency of extension updates increasingly or the test time is greatly extended, the detection method may not be able to meet the testing requirements. Namely, the framework cannot be automatically adjusted in time according to the extension changes.

The future work mainly includes two aspects. The first is to improve the applicability of the framework, and it can be automatically adjusted in terms of classification results, test page construction, etc., according to extensions updates, additions, or reductions. We will use machine learning to solve these problems in our future work. The second point is that the previous work was aimed at Chrome browser extensions. As far as we know, there has not been a comprehensive comparison between mainstream Chinese browser extensions and Chrome extensions. And these are the focus of our future work.

Finally, we call on browser manufacturers to protect users' legitimate rights and interests, and the application store has an inevitable responsibility. On the one hand, it should enhance the detection process before the extension is released. Extensions with malicious behavior should be prohibited from being released. On the other hand, we strongly recommend that the application store force developers providing privacy policy in detail, which explains which servers the private data transmitted to and what data the extension collected, and warn all users that all unofficial extensions may have disclosure behavior.

REFERENCES

[1] A. Barth, A. P. Felt, P. Saxena, and A. Boodman, "Protecting browsers from extension vulnerabilities," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, USA, 2010, pp. 1–13.

[2] A. Salih Buyukkayhan, K. Onarlioglu, W. Robertson, and E. Kirda, "CrossFire: An analysis of firefox extension-reuse vulnerabilities," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2016, pp. 1–12.

[3] A. Castiglione, M. Nappi, and C. Pero, "Towards the design of a covert channel by using Web tracking technologies," in *Proc. Int. Conf. Dependability Sensor, Cloud, Big Data Syst. Appl.* Singapore: Springer, 2019, pp. 231–246.

[4] W. Chang and S. Chen, "ExtensionGuard: Towards runtime browser extension information leakage detection," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Oct. 2016, pp. 154–162.

[5] Q. Chen and A. Kapravelos, "Mystique: Uncovering information leakage from browser extensions," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 1687–1700.

[6] CNCERT/CC. *Analysis on the Proportion of Operating System and Browser in China*. Accessed: Nov. 18, 2020. [Online]. Available: https://www.cert.org.cn/publish/main/68/2019/20191213093128213770979_.html

[7] CNNIC. *Statistical Report on the Development of Internet in China*. Accessed: Nov. 18, 2020. [Online]. Available: http://www.cnnic.cn/hlwfzyj/hlwxzbg/hlwtjbg/202004/t20200428_70974.htm

[8] P. A. Dev and K. Jevitha, "Stride based analysis of the chrome browser extensions API," in *Proc. 5th Int. Conf. Frontiers Intell. Comput., Theory Appl.* Singapore: Springer, 2017, pp. 169–178.

[9] G. A. Fowler. *I Found Your Data. It is for Sale*. Accessed: Nov. 18, 2020. [Online]. Available: https://www.washingtonpost.com/technology/2019/07/18/i-found-your-data-its-sale/

[10] A. Guha, M. Fredrikson, B. Livshits, and N. Swamy, "Verified security for browser extensions," in *Proc. IEEE Symp. Secur. Privacy*, May 2011, pp. 115–130.

[11] G. Inc. *Chrome Webrequest*. Accessed: Nov. 18, 2020. [Online]. Available: https://developer.chrome.com/extensions/webRequest

[12] G. Inc. *Content Scripts*. Accessed: Nov. 18, 2020. [Online]. Available: https://developer.chrome.com/extensions/content_scripts

[13] G. Inc. *Content Security Policy (CSP) Google Chrome*. Accessed: Nov. 15, 2020. [Online]. Available: https://developer.chrome.com/extensions/contentSecurityPolicy

[14] G. Inc. *Declare Permissions*. Accessed: Nov. 18, 2020. [Online]. Available: https://developer.chrome.com/extensions/declare_permissions

[15] G. Inc. *Manage Events With Background Scripts*. Accessed: Nov. 18, 2020. [Online]. Available: https://developer.chrome.com/extensions/background_pages

[16] G. Inc. *Manifest File Format*. Accessed: Nov. 18, 2020. [Online]. Available: https://developer.chrome.com/apps/manifest

[17] 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 *Proc. Usenix Conf. Secur. Symp.*, 2015, pp. 579–593.

[18] S. Junyi. *Jieba Chinese Text Segmentation*. Accessed: Nov. 18, 2020. [Online]. Available: https://github.com/fxsjy/jieba

[19] A. Kapravelos, C. Grier, N. Chachra, C. Kruegel, G. Vigna, and V. Paxson, "Hulk: Eliciting malicious behavior in browser extensions," in *Proc. USENIX Secur.*, 2014, pp. 641–654.

[20] S. Karami, P. Ilia, K. Solomos, and J. Polakis, "Carnus: Exploring the privacy threats of browser extension fingerprinting," in *Proc. Netw. Distrib. Syst. Secur. Symp.* Reston, VA, USA: Internet Society, 2020, pp. 1–28.

[21] P. Laperdrix, N. Bielova, B. Baudry, and G. Avoine, "Browser fingerprinting: A survey," *ACM Trans. Web*, vol. 14, no. 2, pp. 1–33, Apr. 2020.

[22] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "On the workings and current practices of Web-based device fingerprinting," *IEEE Secur. Privacy*, vol. 12, no. 3, pp. 28–36, May 2014.

[23] K. O'Flaherty. *Data Leak Warning Issued to Millions of Google Chrome and Firefox Users*. Accessed: Nov. 18, 2020. [Online]. Available: https://www.forbes.com/sites/kateoflahertyuk/2019/07/19/data-leak-warning-issued-to-millions-of-google-chrome-and-firefox-users/?sh=510c112b3ded

[24] K. Onarlioglu, A. S. Buyukkayhan, W. Robertson, and E. Kirda, "SENTINEL: Securing legacy firefox extensions," *Comput. Secur.*, vol. 49, pp. 147–161, Mar. 2015.

[25] R. Pardo, P. P. Sanchez, G. Schneider, and J. Tapiador, "A runtime monitoring system to secure browser extensions (extended abstract)," in *Proc. Secur. Princ. Trust Hotspot*, 2017, pp. 30–34.

[26] R. Perrotta and F. Hao, "Botnet in the browser: Understanding threats caused by malicious browser extensions," 2017, *arXiv:1709.09577*. [Online]. Available: http://arxiv.org/abs/1709.09577

[27] G. Pugliese, C. Riess, F. Gassmann, and Z. Benenson, "Long-term observation on browser fingerprinting: Users' trackability and perspective," *Proc. Privacy Enhancing Technol.*, vol. 2020, no. 2, pp. 558–577, Apr. 2020.

[28] S. Rauti and V. Leppänen, "Browser extension-based man-in-the-browser attacks against Ajax applications with countermeasures," in *Proc. 13th Int. Conf. Comput. Syst. Technol. (CompSysTech)*, 2012, pp. 251–258.

[29] G. Rizothanasis, N. Carlsson, and A. Mahanti, "Identifying user actions from HTTP(S) traffic," in *Proc. IEEE 41st Conf. Local Comput. Netw. (LCN)*, Nov. 2016, pp. 555–558.

[30] F. Roesner, T. Kohno, and D. Wetherall, "Detecting and defending against third party tracking on the Web," in *Proc. Usenix Conf. Netw. Syst. Design Implement.*, 2012, p. 12.

[31] I. Sanchez Rola, I. Santos, and D. Balzarotti, "Extension breakdown: Security analysis of browsers extension resources control policies," in *Proc. 26th USENIX Secur. Symp.* Vancouver, BC, Canada: USENIX Association, 2017, pp. 1–17. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technicalsessions/presentation/sanchez-rola

[32] A. Sjösten, S. Van Acker, P. Picazo-Sanchez, and A. Sabelfeld, "Latex gloves: Protecting browser extensions from probing and revelation attacks," in *Proc. 26th Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*. San Diego, CA, USA: The Internet Society, Feb. 2019. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/latex-gloves-protecting-browser-extensions-from-probing-and-revelation-attacks/

[33] O. Starov and N. Nikiforakis, "Extended tracking powers: Measuring the privacy diffusion enabled by browser extensions," in *Proc. 26th Int. Conf. World Wide Web*, Apr. 2017, pp. 1481–1490.
[34] O. Starov and N. Nikiforakis, "XHOUND: Quantifying the fingerprintability of browser extensions," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 941–956.
[35] G. Varshney, M. Misra, and P. K. Atrey, "Detecting spying and fraud browser extensions: Short paper," in *Proc. Multimedia Privacy Secur.*, Oct. 2017, pp. 45–52.
[36] M. Weissbacher, E. Mariconti, G. Suarez-Tangil, G. Stringhini, W. Robertson, and E. Kirda, "Ex-ray: Detection of history-leaking browser extensions," in *Proc. 33rd Annu. Comput. Secur. Appl. Conf.*, Dec. 2017, pp. 590–602.
[37] X. Xing, W. Meng, B. Lee, U. Weinsberg, A. Sheth, R. Perdisci, and W. Lee, "Understanding malvertising through ad-injecting browser extensions," in *Proc. 24th Int. Conf. World Wide Web*, May 2015, pp. 1286–1295.
[38] R. Zhao, C. Yue, and Q. Yi, "Automatic detection of information leakage vulnerabilities in browser extensions," in *Proc. 24th Int. Conf. World Wide Web*, May 2015, pp. 1384–1394.
[39] Y. Zhao, L. He, Z. Li, L. Yang, H. Dong, C. Li, and Y. Wang, "Large-scale detection of privacy leaks for BAT browsers extensions in China," in *Proc. Int. Symp. Theor. Aspects Softw. Eng. (TASE)*, Jul. 2019, pp. 57–64.

YUFEI ZHAO is currently pursuing the Ph.D. degree in computer science and engineering with Beihang University. His current research interest includes privacy-preserving data mining.

LIQUN YANG is currently pursuing the Ph.D. degree in computer science and engineering with Beihang University, Beijing, China. He is also a Visiting Student with the Department of Electrical and Computer Engineering, Mcgill University, Canada. His research interests include industrial control systems security and cybersecurity.

ZHOUJUN LI received the Ph.D. degree from the National University of Defense Technology, Hunan, China, in 1999. He is currently a Professor with the School of Computer Science and Engineering, Beihang University, Beijing, China. His main research interests include concurrency theory and process algebra, formal analysis and verification of security protocols, information security, and data mining.

LONGTAO HE was born in China, in 1974. He received the B.E., M.S.E., and Ph.D. degrees from the Harbin Institute of Technology, China. He is currently a Professor of engineer with the National Computer Network Emergency Response Technical Team/Coordination Center of China (CNCERT). His research interests include Internet measurement, network traffic classification, Internet security, and mobile security.

YIPENG ZHANG is currently pursuing the Ph.D. degree in computer science and engineering with Beihang University, Beijing, China. His research interests include web security and ICS security.