

Metadata Analysis of Popular Browsers

Jack Bowker – 1803838

Metadata Analysis of Popular Web Browsers

CMP320 Ethical Hacking 3

2019/20

Abstract

Analysis was done on Google Chrome, Mozilla Firefox, UC Browser and Un-googled Chromium to measure metadata and what data each browser sent while in use, recording HTTPS traffic using MITM proxy on Windows.

This analysis covered the Installation and first launch of each browser, including any identifiers that persisted through reinstallation, Letting the browser sit idle, and the default settings each browser had enabled.

It was found that as expected Un-googled Chromium was the most privacy friendly browser, not sending any data at all throughout all the tests as well as not using Safe Browsing features or being able to update by default. However, it was found to be unfriendly for new users due to the lack of a search engine by default.

Firefox was the second most privacy friendly, although it sent telemetry data by default it could be turned off, as well as having good options to block cookies and trackers, as well as offering DNS over HTTPS by default. Overall, it was found to be a flexible option suitable for new users as well as power users who would be comfortable changing a wide range of configuration options.

Google Chrome closely followed behind Firefox as it also had telemetry features, it also offered the option to use DNS over HTTPs as well as having good documentation to find out what data collected was used for. Chrome is the benchmark for modern browsers and there's a reason it has such dominant market share, although it could be made more privacy friendly if it offered different search engines up front as well as options to disable telemetry.

In far last place, UC browser had no public documentation, as well as sending unreadable encrypted data to servers in China even when the application was closed. The browser had a reasonable set of options for ad-blocking built in but due to the constant sending of unknown data it can't be recommended.

Contents

Abstract	2
Introduction	4
Procedure	4
Testing Environment	4
Browser Selection	4
Methodology	5
Installing and first launch of each browser	5
Letting the browser sit idle	6
Comparing privacy defaults	6
Results	6
Installing and first launch of each browser	6
Google Chrome	6
Mozilla Firefox	8
Un-googled Chromium	9
UC Browser	10
Letting the browser sit idle	12
Google Chrome	12
Mozilla Firefox	12
Un-googled Chromium	13
UC Browser	13
Comparing Privacy Defaults	14
Google Chrome	14
Mozilla Firefox	15
Un-Googled Chromium	16
UC Browser	16
Discussion	17
Conclusions	17
Future Work	17
References	18

Introduction

More and more applications are moving from being native running on Windows, to being web-based, meaning the Web Browser is more than ever almost a secondary operating system on PCs. Browsers deal with all sorts of sensitive information – from online banking, learning, social media and many other personal purposes.

This importance makes it crucial to have an idea of what browsers are doing while they're open, as well as what data they send while in use.

The following report will analyse popular browsers from different areas of the market – From the most mainstream western browsers such as commercial Google Chrome and open source Mozilla Firefox, to more obscure browsers such as UC Browser – which is very popular in the Asian market, and Un-googled Chromium which is a more technical user looking to send as little data as possible.

Procedure

Testing Environment

It was important during the analysis of the browsers to have a consistent environment to carry out testing. For example, it's possible if using an existing OS installation, that a user could have configured custom privacy/security settings that aren't cleared on a program's removal. Therefore, the decision was made to use a Virtual Machine with a new Operating System installation.

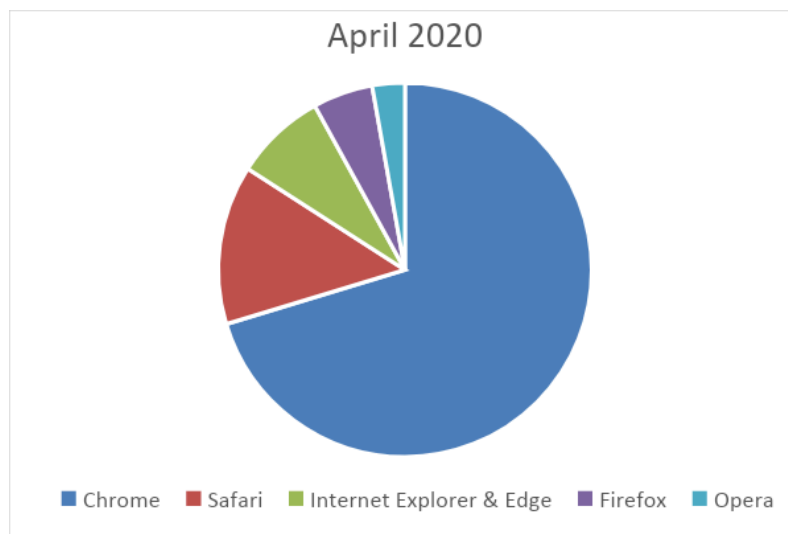
A fresh installation of Windows 10, version 1909, was installed on VMware Workstation. Windows was chosen because it makes up the vast majority of Operating System market share, along with the fact that it supports most modern mainstream browsers. Although due to the increasing amount of internet browsing done via mobile devices, Windows was chosen due to the ease of monitoring and recording what the browsers were doing.

The "MITM proxy" tool was used to track connections that the web browsers were making. This tool comes with both a command line interface and a web UI that allows for the monitoring, manipulating and saving of network data. Along with this, it included a root certificate which allowed for sniffing of HTTPS traffic. Once configured, this proxy was selected in Windows settings so all traffic would flow through it.

Browser Selection

When deciding which browsers would be part of this analysis, it was important to include a range of browsers with different philosophies on privacy. This meant including browsers that both promoted the fact that they held no data on their users, as well as browsers from more arguably shadier companies with a bad track record on tracking their users.

Shown below are the W3Counter metrics for global browser usage in 2020. (W3Counter, 2020)



As shown above Chrome has the vast majority of desktop browser market share, so it was important to represent this in testing. Chrome's lead is added to due to the fact the open source Chromium browser is also used in the most recent release of Microsoft Edge, as well as the Opera browser.

Safari is the second most popular browser according to W3Counter. This can partly be attributed to it being the default choice on Apple's Mac computers, and unfortunately won't be tested in this scenario since the Windows version of the browser was abandoned in 2010 with Safari version 5 being the final release.

The decision was made to depart slightly from the most popular browsers with the inclusion of UC Browser, as it has over 500 million users in mostly developing countries, and Un-googled Chromium as a contrast to show that the Chromium framework can be used in a privacy-conscious way.

Methodology

In order to reliably compare the data different browsers sent, it was important to have a set methodology that could be reproduced in different scenarios. The following tests were carried out whilst recording network traffic generated.

Installing and first launch of each browser

This step involves going through the built-in installer for each browser using default options, then leaving the browser open for a minute once installed. This showed what different analytics and identifiers are sent, along with what is downloaded on the first launch of the browsers.

In attempt to simulate an average user, any tick boxes that were checked by default weren't changed – for example, any tick-boxes to "Automatically send usage statistics and crash reports" were kept ticked.

After the browser was installed, it was launched if it wasn't automatically and left to idle for approximately 1 minute. This was to see if any machine identifiable identifiers were sent during the setup process.

The browsers will then be reinstalled, to check for any persistent unique identifiers that are tied to the machine instead of the browser install.

Letting the browser sit idle

The browsers will be left for 30 minutes, idle on the virtual machine. This will help measure any data that is transmitted on a schedule in the background while the browser is open.

Comparing privacy defaults

Default settings will be compared throughout the four browsers and will be compared based on how they deal with the default settings for things such as search engine selection, cookies, and other features.

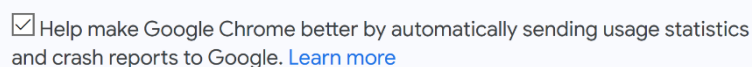
Results

Installing and first launch of each browser

Google Chrome

Installation

This was the first browser to be installed on the virtual machine. When downloaded using the “Download Now” button on Google’s website, the optional telemetry was enabled as shown below and this was left ticked. A web installer was then downloaded, this being a minimal package which will ping Google’s servers to fetch the latest browser version, then downloading the full browser and running through the installation of it.

A screenshot of a checkbox during Google Chrome installation. The checkbox is checked, and the text reads: "Help make Google Chrome better by automatically sending usage statistics and crash reports to Google. [Learn more](#)".

☒ Help make Google Chrome better by automatically sending usage statistics and crash reports to Google. [Learn more](#)

Throughout the installation, 5 unique identifiers were sent using a POST request, to the URL “https://update.googleapis.com/service/update2”.

```
sessionId="{F4143ED1-D51A-4C63-A98E-674CC79D3188}"
userid="{8A3181E3-EB38-4FDB-BE24-7AB0493503BB}"
requestid="{7D3EFB7A-77C8-460B-BA9F-6BB7680CC07B}"
appid="{430FD4D0-B729-4F61-AA34-91526481799D}"
iid="{4FA4D8C9-4D45-6083-A8B4-31511E405BD4}"
```

Along with this, some basic hardware capability information was sent along with the version of Windows running on the virtual machine.

```
<hw physmemory="4" sse="1" sse2="1" sse3="1" ssse3="1" sse41="1" sse42="1" avx="1"/><os platform="win"
version="10.0.18363.418" sp="" arch="x64"
```

This request was responded to with a direct URL target for the full Google Chrome browser executable as well as hashes to verify the download. Next, a “verbose-logging” flag was found in the response which likely corresponds to the “usage statistics” box that was checked by default on the browser download page.

```
<action arguments="--verbose-logging --do-not-launch-chrome" event="install"
run="81.0.4044.138_chrome_installer.exe"/>
```

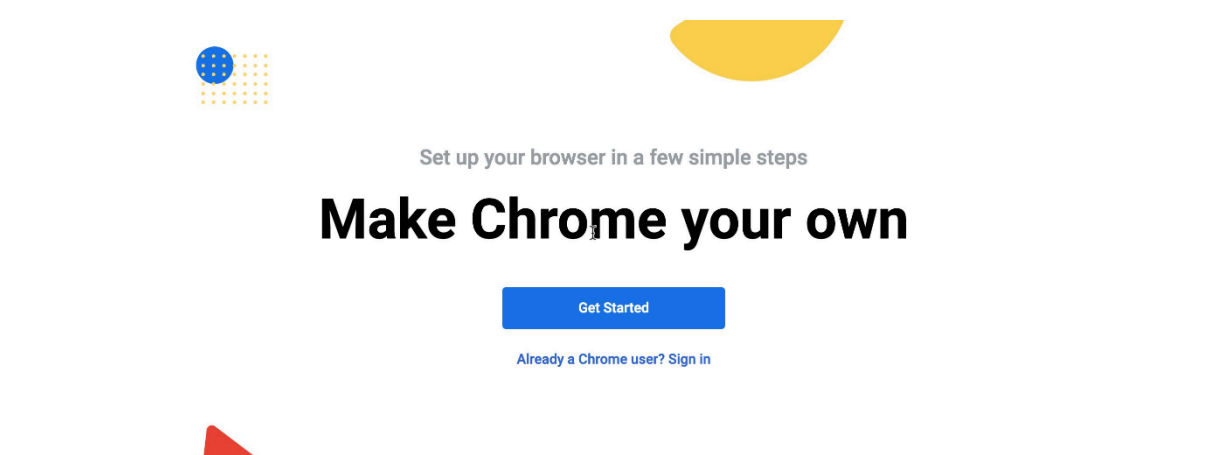
Finally, a “variations_compressed_seed” was transmitted, which according to Google Chrome’s documentation is a randomly generated number used in field testing. (Google, Chrome Variations, 2020)

```
{"variations_compressed_seed": "H4sIAAAAAAAAA\+/y9CZgk11UmWp1VJ0
NgzGCYxQvgMQbb72GewCAWYzbPDKsfzQAxMMOAsbFZ\//L6I3CIyI5fqbtmSXP
tn7IZjrTStHdvnuhIh3HchsKsjCKpBaNwadhUCUGjUDo0EKkNB1Dw39L1mkmw
/yhQlatwUDiq1VuBcto2nt23fsd26bbjrn1+3ea78CuFGw42KaBQatYXRpJUT
woFOkVXdqdtnnA0\//oYKDXRQuGHLdJe9hnhzw6nb1a3ux9rtgjiMW7ldnKN
```

Using the browser

Once Google Chrome had been installed and launched for the first time, the first call the browser made was to “accounts.google.com” and appears to be trying to find Google Accounts that are already linked to this device. However, in this case the GAIA (Google Accounts and ID Administration) (Google, GAIA Access Control, 2020) response was null as this VM doesn’t have any other Google services installed.

```
[
  "gaia.l.a.r",
  []
]
```



After querying for a Google Account, the browser checks the Web Store for extension updates for the default Chrome extensions (from first to last - Slides, Docs, Google Drive, YouTube, Sheets, Docs Offline, Chrome PDF Viewer, Gmail and Chromecast).

x-goog-update-appid

aapocclcgogkmnckokdopfmhnmfmgok, aohghmighlieiainnegkcijnfilokake, ap
df1lckaahabafndbhieahigklhalf, blpcfgoakmgknkcojhhkbfbldkacnbeo, felc
aaldbndnclmgdcncolpebiejap, ghbmnnjoekpmoecnnnilnnbdloihkhi, nmmhkk
egccagdldgiimedpiccmgieda, pjkljhegncpnkpknbcohdijeoejaedia, pkedcjkd
efgpdelpbcmbeomcjbemfm

This check is successful, and Google responds to notify all extensions are up to date with the message “status=ok”.

Once extensions are updated, Chrome queries

“https://www.gstatic.com/chrome/intelligence/assist/ranker/models/translate/” for a binary file, which according to Chromium source code will “decide whether or not Translate UI should be triggered in a given context”.

Finally, the browser queries “https://ssl.gstatic.com/safebrowsing/” as well as “https://safebrowsing.googleapis.com/v4/threatListUpdates” to fetch safe browsing data and blocklists.

Overall, 2.3MB of data was transmitted during the installation and first launch of the browser, excluding the browser binary being downloaded.

Mozilla Firefox

Installation

Mozilla Firefox also uses a web-based installer to download and install the browser and utilities. However, unlike Google Chrome there was no usage statistics check box to opt-out of.

Once the browser executable was downloaded, the browser prompts for installation of the “Mozilla Maintenance Service”. According to Mozilla’s site, it’s installed as a service, so the user doesn’t have to dismiss UAC (User Access Control) warnings when the browser updates. (Mozilla, What is the Mozilla Maintenance Service?, n.d.)

The first non-download related request was to “download-stats.mozilla.org” which reported the downloaded browser version along with the referral link used to find the download page, in this case Bing on Microsoft Edge. The version statistics are publicly available on the Firefox Data dashboard. (Mozilla, Firefox Public Data Report, n.d.)

GET

http://download-stats.mozilla.org/stub/v8/release/release/en-GB/1/1/10/0/18363/0/0/0/2/0/52306272/52306272/0/0/7/6/0/1/8/0/0/0/0/0/0/76.0.1/20200507114007/1/1/0/1/Unknown/campaign%3D%2528not%2Bset%2529%26content%3D%2528not%2Bset%2529%26experiment%3D%2528not%2Bset%2529%26medium%3Dreferral%26source%3Dwww.bing.com%26ua%3Dedge%26variation%3D%2528not%2Bset%2529/0/0 HTTP/1.1

Using the browser

After this, Firefox opened for the first time and a “captive portal” page was pinged with 18 requests for both IPv4 and IPv6. This feature is made to detect login portals for things such as paid Wi-Fi, and it’s possible that either the MITM proxy or VM networking could have gotten it stuck for a while.

http://detectportal.firefox.com/success.txt	GET	200	8b
http://detectportal.firefox.com/success.txt?ipv4	GET	200	8b
http://detectportal.firefox.com/success.txt?ipv6	GET	200	8b
http://detectportal.firefox.com/success.txt?ipv4	GET	200	8b
http://detectportal.firefox.com/success.txt?ipv6	GET	200	8b
http://detectportal.firefox.com/success.txt	GET	200	8b
http://detectportal.firefox.com/success.txt?ipv4	GET	200	8b
http://detectportal.firefox.com/success.txt?ipv6	GET	200	8b
http://detectportal.firefox.com/success.txt	GET	200	8b
http://detectportal.firefox.com/success.txt?ipv4	GET	200	8b
http://detectportal.firefox.com/success.txt?ipv6	GET	200	8b
http://detectportal.firefox.com/success.txt	GET	200	8b
http://detectportal.firefox.com/success.txt?ipv4	GET	200	8b
http://detectportal.firefox.com/success.txt?ipv6	GET	200	8b
http://detectportal.firefox.com/success.txt	GET	200	8b

After this, a GET request is sent to

“https://firefox.settings.services.mozilla.com/v1/buckets/monitor/collections/changes/records?collection=message-groups&bucket=main”, which fetches a list of harmful add-ons collated by Mozilla which will stop them from being installed.


```
POST
https://incoming.telemetry.mozilla.org/submit/messaging-system/onboarding/1/2de10614-c7b7-4be6-b21e-2ea3966655fa HTTP/1.1

{"browser_session_id": "8ebd1035-818f-4648-9bb4-2824a2327f73",
 "client_id": "64b69459-c8c9-4569-ada0-2ec6dd29d5ce",
 "locale": "en-GB",
 "release_channel": "release",
 "version": "76.0.1"}
```

Origin	wss://push.services.mozilla.com/
Sec-WebSocket-Protocol	push-notification
Sec-WebSocket-Extensions	permessage-deflate
Sec-WebSocket-Key	VRlfy2BT260JgmMBE3Rbm==

/WINNT_x86_64-msvc/en-GB/release

```
-----BEGIN CERTIFICATE-----
MIIDBCCAouAgIBAgIIFg1m4AXws6UwGyYIKoZiZjOEAnhwgAxBzA7BgNVBAYT
A1VTNmRmGyQVODQXENWb3ppbGxhIEVncnBvcnR0aW9uS1BwLjQVYDQVQVYENb3pp
bGxhIEVudF9Yb2Qmcm9kaW9uIFNpZ2BpcmcuZ2Vudm1jZTFFMENGAIUEAnw8Q29u
dG9uZC9BTAludkudS1EldUwGvblWkaf0ZS91blB0FkFkZDJ1C3M9Zm94c2VlIG1u
```

```
"deviceId": "5f9571da1de748248efbfbff5d3541d",
"flowBeginTime": 1589548978001,
"flowId": "ee93ff48c89c910c3d9e990e4b59f77640e4cc8333cd6e3ffddf5b91b49bf8d"
```

Overall, Firefox transmitted much less data than Google Chrome but the large Safe Browsing list (4.2MB) brings the total to 4.4MB.

Compared to the other browsers in this test, this browser was definitely the most geared towards privacy conscious users.

The development and downloads are handled in GitHub, with prebuilt binary downloads being available for multiple platforms. The latest version available for Windows at the time of testing was “80.0.3987.149”, as since it’s an open source project and doesn’t have a dedicated development team it often falls behind the latest Chromium version.

After downloading the installer executable, the program created no network traffic at all throughout the installation process.

Using the browser

Once installed the browser was launched, and there was no traffic generated as the browser doesn’t check for updates automatically, along with not using Safe Browsing lists or updating extensions through the Chrome Web Store. Along with this, the default launch page is an offline page showing recently visited sites so no network queries are required.

UC Browser

Installation

Unfortunately, even after tweaking with MITM proxy configuration the UC browser installer refused to traffic through the proxy, so network activity data only starts from when the browser was launched for the first time.

This should be considered when measuring network data as it’s possible the browser was working around the proxy even once open.

Using the browser

The first request from UC made appeared to be querying the IP address of the machine, with the URL as following over an unsecure HTTP connection.

```
http://ucip.uc.cn/get_ip_attr?type=1&format=0&caller=gj_pcbrowser&key=097a6150b0c772f7952807c0cb48fb86
```

The server responded with what looked vaguely like a WHOIS lookup for something in Columbus, Ohio in the United States.

```
{"respcode": "1000", "ccode": "US", "country": "美国", "isp": "", "province": "OH", "city": "Columbus", "zcode": ""}
```

Along with this, when the browser sat idle it sent constant GET requests to “i18nmmstat.ucweb.com” which consistently started with the string “encrypt_data=bTkWAmL569”, continuing with seemingly random characters as shown below.

https://i18nmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAb6A8WIMwCeyBJt3C+nmXP780TYbb... GET	200
https://i18nmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAmj8y5xgDU9dJhCl550jCSVL67MfclZfv/hn... GET	200
https://i18nmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAnkAPc1xCdkMFRQ31PW7LUOp41e9Q/rx... GET	200
https://i18nmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAnC7bdzCyk2F+qHyvG3TW+H/dJXvydTK... GET	200
https://i18nmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAnC7bdzCyk2F+qHyvG3TW+H/dJWfBTq... GET	200
http://ucip.uc.cn/get_ip_attr?type=1&format=0&caller=gj_pcbrowser&key=097a6150b0c772f7952807c... GET	200
https://i18nmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAiII6DfAedZY+xdN4yml2T3d8wMXQu7ia... GET	200
https://i18nmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAiII6DfAedZY+xdN4yml2T3d8wMXQu7ia... GET	200
https://i18nmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAmcO41jAd9MJ+w1UJGLKcNF81M9Dumz... GET	200
http://www.uc123.com/guide/install_blacklist.php?ver=6.0.1308.1016&bid=35151&pid=4601&mid=105... GET	200
https://i18nmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAjCJRcyzAsENV+Mv13GJHUUn8Ue/ntJD2... GET	200
https://i18nmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAhkNQZzRHsVddec5zWBFTv01kAae96hv... GET	200
https://i18nmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAmEJS7uZAs8yPeMlxqWJCWdb8bNBHNI... GET	200
https://i18nmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAhsNMq7XHtQve+c69TmBOWHvyWF7btM... GET	200
https://i18nmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAhsNMq7XHtQve+c69TmBOWHtyWF7atM... GET	200
https://i18nmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAhsNMq7XHtQve+c69TmBOWHsyWF7atM... GET	200
https://i18nmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAlcOabeTH602N+YDyrGfRW9b49JN9B1... GET	200

The next different request was a GET request sent to “uc123.com” with the URL “install_blacklist.php” followed by the browser version and some unique identifiers.

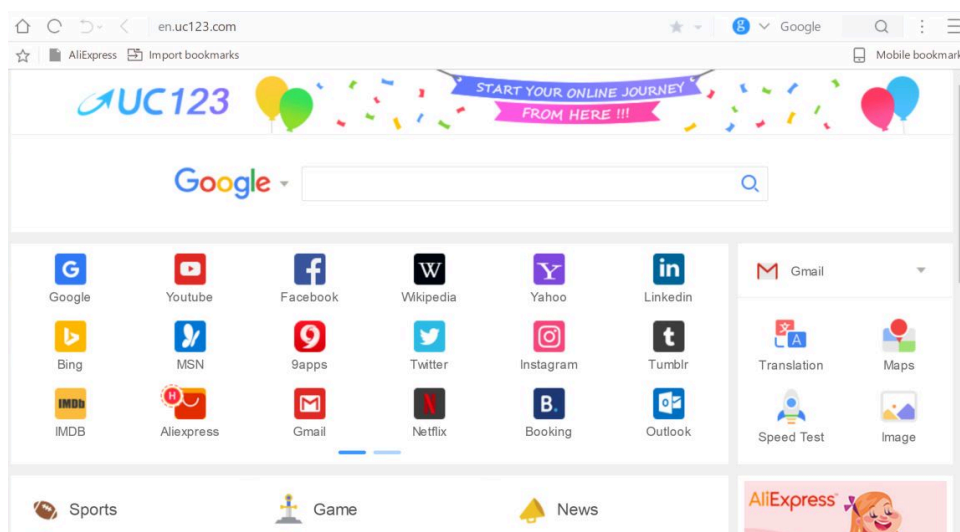
```
ver=6.0.1308.1016
bid=35151
pid=4601
mid=10586eef3962632c1c4213dc25213ed8
midex=7df2aecd1257f373c64dafab3205c5d2v00000023084c5a4
```

The server responded with the message “no way!” as shown below.

no way!

From here, “gj.track.uc.cn:9080” was contacted in what appeared to be a request for the new tab page displayed on launch of the browser as shown below. The same unique identifiers were transmitted, however this time “midex” is referred to as “uuid”. As shown, this is also sent over an insecure HTTP connection.

```
http://gj.track.uc.cn:9080/collect?pg=newtabpv&lt=event&appid=2796a51d9ed0&fr=PC&ver=6.0.1308.1016&uui
d=7df2aecd1257f373c64dafab3205c5d2v00000023084c5a4&firstpid=4601&bid=35151&lang=en-US&reload=0
```



After loading the new tab page, “browser.taobao.com/extensions/update.htm” is contacted to update an extension with the ID “hkmogefbfdbmoplojeicpibfpcndjjbm”.

```
<app appid="hkmogefbfdbmoplojeicpibfpcndjjbm">
  <updatecheck codebase="https://wow.uc.cn/extensions/787a5a66-e0da-4ff8-ab86-f0a52b67b26c.crx"
version="1.7.3" />
</app>
```

When investigated further, this appears to be the UC Fast Video Downloader, and when the CRX file was extracted it was found to automatically inject itself on a multitude of video hosting and pornographic websites.

```
content_scripts":[{"matches":["*://v.youku.com/v_show/*"],"js":["inject/youku.js"],"css":["inject.css"],"run_at":"document_end"},
{"matches":["*://vimeo.com/*","*://www.xvideos.com/*","*://*.tube8.com/*","*://*.redtube.com/*","*://www.dailymotion.com/video/*"
*","*://www.youporn.com/*/*","*://www.gaytube.com/media/*","*://www.thumbzilla.com/video/*","*://www.pornhd.com/videos/*","*://ww
["inject/video-injector.js"],"run_at":"document_start"},{"matches":["*://*.music.qq.com/*","*://y.qq.com/*","*://music.qq.com/*"]
```

Finally, UC Browser contacted the same Safe Browsing page as chrome at “ssl.gstatic.com/safebrowsing/” to download safe browsing data.

Reinstalling the browser

Unlike every other browser in this test, after fully reinstalling the browser and deleting browsing data from “%AppData%\UC Browser” there were still identifiers found during usage that matched the identifiers from the first installation.

These were found in the “install_blacklist.php” request sent to “uc123.com”, similarly to as was sent after the first installation.

```
http://www.uc123.com/guide/install_blacklist.php?ver=6.0.1308.1016
&bid=35151
&pid=4601
&mid=10586eef3962632c1c4213dc25213ed8
&midex=7df2aecd1257f373c64dafab3205c5d2v00000023084c5a4 HTTP/1.1
```

This indicates that UC browser is identifying machines instead of the browser installation, either by generating these IDs based on fingerprinting methods or a hidden file that is generated the first time UC browser is installed, and means that even if all cookies and browsing data are deleted, if the browser is reinstalled it can identify an individual user.

Letting the browser sit idle

Google Chrome

Whenever opened, Chrome will contact “accounts.google.com/ListAccounts” to check if the user is signed in, so it can prompt to sync bookmarks and history. This was the first request made but since no Google accounts were found it returned the same null value as it did during installation.

After this, the new tab page queried “google.com/async/newtab_ogb” to fetch the style, content and JavaScript to be displayed on the new tab page. Along with this, promos were queried, as occasionally on the new tab page offers on the Google Store are displayed here.

Apart from this, Chrome checked for updates to the Safe Browsing list, extension updates, and updates to the stable branch of the browser. In total, 312kb of data was transmitted over the 30 minutes the browser was idle for.

Mozilla Firefox

The first requests made when Firefox opened were the captive portal requests, which returned “successful” after sending a small GET request to “detectportal.firefox.com”

From here, the blacklisted extension list is queried and updated from the “firefox.settings.services.mozilla.com” domain.

Next, Firefox sends some telemetry data similar to during installation where it sends an “impression_id”, which was different to the one seen during installation, along with locale and browser version.

```
{
  "addon_version": "20200507114007",
  "event": "ASR_RS_NO_MESSAGES",
  "event_context": "message-groups",
  "experiments": {},
  "impression_id": "{649ec9f6-d627-4810-9566-67882a63e27b}",
  "locale": "en-GB",
  "message_id": "n/a",
  "release_channel": "release",
  "version": "76.0.1"
}
```

After that, Firefox queries for extension updates to the preinstalled OpenH264 and WidevineCDM media codecs with the domain “aus5.mozilla.org”, and then downloads an update for Widevine from “gvt1.com”, which is a Google server.

Finally, the browser queries for updates to content signatures and safe browsing similarly to when installed.

Overall, Firefox transferred 5.1MB during the 30 minutes of idle time, however 4.9MB of this was a Widevine update.

Un-googled Chromium

Once opened, no network connections were made throughout the 30 minutes the browser was left open, and 0 bytes were transferred.

UC Browser

Every time the browser is opened it triggered a few tracking requests on both “gj.track.uc.cn:9080” and “ip.taobao.com/service/getIpInfo.php”, with the Taobao request returning the response message “the request over max qps for user ,the accessKey=public”

Similarly to the other browsers, after being reopened UC queried for updates to extensions, as well as the safe browsing list and for any updates to the browser itself.

After a few minutes of the browser being open, it sends a request to “pcus.ucweb.com” with some encoded data appearing to consist of browser settings and personal data as shown below:

```
\x0bpc_bookmark\x12\x0856f1d50a"!
\x15pc_cloud_acceleration\x12\x0894ca4c8b"(
\x1cpc_cloud_acceleration_effect\x12\x085c252bb5"\x15
pc_custom\x12\x0803710af2"\x17
\x0bpc_download\x12\x08cecc68f5"\x1c
\x10pc_engine_switch\x12\x0884bc0bbb"\x18
\x0cpc_extension\x12\x08a22e8d87"\x1e
\x12pc_feature_control\x12\x08a19aed5b"\x1f
```

```

\x13pc_image_accelerate\x12\x0875b6c44c"\x14
\x08pc_login\x12\x0816024b04"\x13
\x07pc_misc\x12\x0881d9efa7"\x1a
\x0epc_newtab_gird\x12\x084a32166f"\x1a
\x0epc_newtab_grid\x12\x08eb5f5441"$
\x18pc_newtab_recommendation\x12\x08f52428ce"!
\x15pc_preset_url_library\x12\x08704516bd"%
\x19pc_private_api_permission\x12\x08abe622ba"\x1c
\x10pc_render_engine\x12\x080dab7f2c"
\x14pc_resource_prefetch\x12\x08b3457a6c"\x1c
\x10pc_search_engine\x12\x08f9ed37e0"\x19

pc_share_misc\x12\x08d5537938"&
\x1apc_startup_feature_control\x12\x083bfafc76"\x1a
\x0epc_startup_gpu\x12\x08a2a1780e"\x1b
\x0fpc_startup_home\x12\x085b378daf"\x1b
\x0fpc_startup_misc\x12\x08995ca92e"\x19

pc_url_handle\x12\x085965e13b"\x17
\x0bpc_url_list\x12\x0844cfa128"\x0c
\x04null\x12\x04nullH\x00R.isp.;prov:OH;city:Columbus;na:\xe7\xbe\x8e\xe5\x9b\xbd;cc:US;ac:

```

From here, the queries slowed down except from the same consistent requests to “i18nmmstat.ucweb.com”, which continued every 10 or so seconds the browser was open.

```

GET
https://i18nmmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAizHhPykQIb9ai3sBxsNn+UxdcXfvSpYnYYT09zPIBgG9r2Hsm4770Le+yxv0Z2opqyaSCd7kyn5uvL977HfaswJ7RyQ1YCgAmTcvFovv0VFU2DRBsq0Bfd6ey/IIPzGrzUamaJA9M+diyNruuiHUEzhM1onDp4vUotc9Sqsne32l+ASx2D14bv95uzNV6d7y0GNhhiqtzQbZvVyH0Tks1wI+5Ivp0tLrAvbMfjMSVmmYphd9DBLuqnVZssgsjiUXqNae1dD0hcSYA==

```

The MITM proxy was left to idle with none of the browsers open, and worryingly the requests to “UCWeb” continued once the browser was closed as shown in the image below.

```

https://i18nmmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAizHhPykQIb9
https://i18nmmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAibHuuSQIziI
https://i18nmmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAh/II3jbQedxfy
https://i18nmmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAiLIRGTeQcZI
https://pcus.ucweb.com/usquery.php
https://i18nmmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAl/Jq0ebQm+C
https://i18nmmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAie9hP+jTob+;
https://i18nmmstat.ucweb.com/lv=1.0&encrypt_data=bTkWAnq9z9Z2Tks)

```

Comparing Privacy Defaults

The default settings of each browser will be analysed to compare the level of privacy each browser provides the user by default. The comparison will be based on the following main sections:

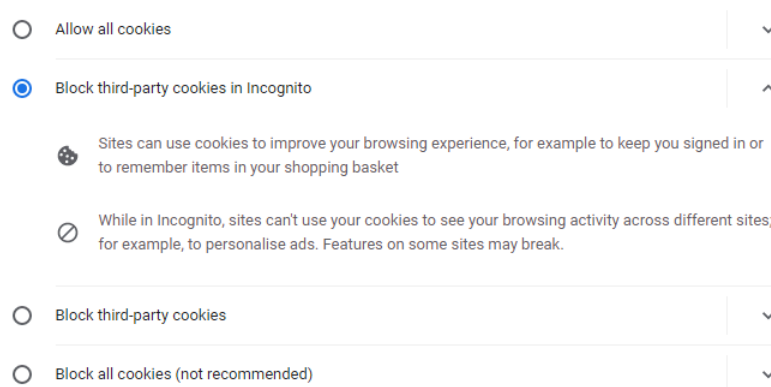
- Default search engine & Autocomplete
- Enhanced Telemetry
- Cookie handling
- DNS Security

Google Chrome

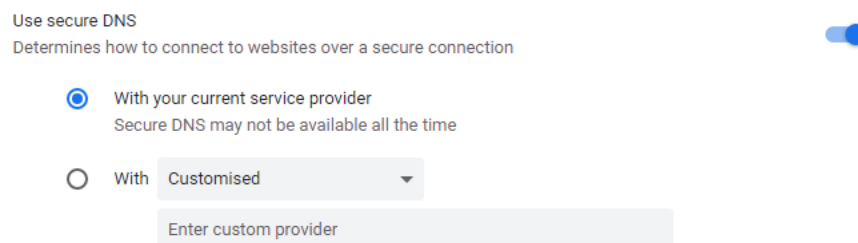
Google Chrome by default comes with Google Search enabled as well as autocomplete. The browser gives other options such as DuckDuckGo and Bing but these are not able to be changed without going into browser settings.

By default, as shown above in the Installation section, Google opts the user in to “Automatically sends usage statistics and crash reports to Google”. This will send details such as memory dumps (which could include passwords or other sensitive data), Chrome settings, Extensions, and the website that crashed Chrome.

Third party cookies are enabled by default in Chrome, with them being blocked only in Incognito unless changed.



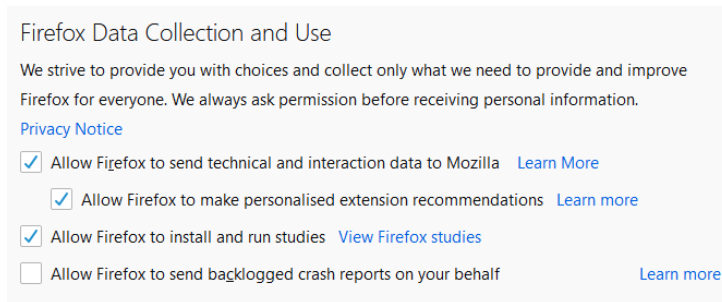
Google Chrome also offers DNS security, however it's only enabled if the default upstream DNS provider (for example BT) enable it first, which is rare. It does however allow the user to input a custom DNS over HTTPS server to use



Mozilla Firefox

Firefox, like Chrome, comes with Google Search enabled as well as autocomplete by default. The main difference between the two browsers is that Firefox allows the user to easily switch search engines from the address bar.

Firefox similarly to Chrome will send usage statistics automatically, but unlike Chrome will not send crash reports containing potentially sensitive data by default. Along with this, the browser gives the option to disable all telemetry using toggles in settings along with more granular control in the browser flags.



The major difference between Chrome and Firefox is that Firefox uses “Enhanced Tracking Protection” which blocks Social media trackers, Cross-site tracking cookies, Cryptominers, along with Fingerprinters all by default. This blocks a lot of third-party cookies but it’s not an all-out blocker, as it instead blacklists known tracking cookies.

Firefox is the first browser to roll out DNS over HTTPS for all users, although however this isn’t by default in all regions yet. The user gets the choice to either use the ISP DNS, or a HTTPS secured DNS service from Cloudflare, Google or a custom address. (Mozilla, Firefox continues push to bring DNS over HTTPS by default for US users, 2020)

Un-Google Chrome

By default, Un-googled Chromium comes with no search engine, and the address bar can only be used to type in URLs. Search engines are available in settings in the same place as Chrome, but instead of Google Search, “Searx” and “DuckDuckGo” are the two top engines as they are both very privacy oriented.

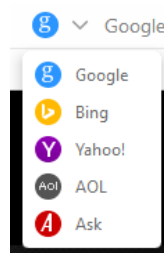
This browser has 0 usage or crash data sent by default, and there’s no options in settings to enable this as the issues are managed and submitted directly on GitHub.

Cookies in Un-googled Chromium are not stored permanently by default, as the “Clear cookies and site data when you quit Chromium” setting is selected by default.

The only disadvantage for Un-googled Chromium is because it’s a few versions behind regular Google Chrome, it doesn’t offer any DNS over HTTPS options in the browser. This will likely be added in the future once the Chromium version is updated to 83.

UC Browser

UC comes with Google as the default search engine with autocomplete on, but as shown below other providers are available easily without having to go into the browser’s settings.



Similarly to Chrome, UC opts the user into the User Experience Improvement Plan by default which sends data such as memory dumps as well as usage information.

Third party cookies are enabled by default on UC browser, and the option to block them is hidden within 3 settings menus. Aside from this, UC is the only browser to have an explicitly stated “Ad blocker” enabled by default. UC has an official filter list for ads to block, but the user can add their own to blacklist certain domains.

There was no DNS options in this browser to enable DNS over HTTPS, and no indication was found that this feature is going to be added in the near future as this browser is not Chromium based.

Discussion

Conclusions

The four browsers fell into 3 levels of privacy protection – Un-googled Chromium by far was the most private browser, followed by Mozilla Firefox and Google chrome, then finally UC Browser in far last.

Un-googled Chromium sent no data throughout the entire testing progress, along with not saving cookies or using any Safe Browsing lists. This could make it the best option for very privacy aware users, however it's not without drawbacks.

Because of the browser's strict philosophy on privacy and not transmitting any user data, it is not as secure as, for example, Firefox or Chrome. This is due to the lack of Safe Browsing utilities and also because it's outdated software, leaving the end user open to vulnerabilities that have already been patched in the other browsers. There is methods to build the browser from source or patch newer versions, but at that point it's unreasonable for any average user to be able to use.

Firefox and Chrome are the two mainstream western browsers in this test, and the results weren't unexpected in that they sent some telemetry data but also chose sensible defaults, with Firefox edging out Chrome in reducing trackers and also being entirely open source. The main offense for Chrome being that crash reporting and tracking cookies are both enabled by default, making it worse that crash reporting is very hard to disable as it's not a visible setting and instead a flag that's enabled when launching the executable.

UC Browser is by far the least privacy friendly browser of the test, and it's very concerning that unreadable data is sent to unknown servers for unknown reasons, as although it could be usage data there is no way to tell and there's no public documentation about the browser unlike Chromium, Chrome or Firefox. As well as this, since the browser is built and the datacentres are within China, there are also concerns about what is done with this data once it's transmitted.

Apart from this, the browser does have adblocking by default unlike any of the browsers in this testing scenario.

Future Work

In the future, it would be very interesting to look further into UC Browser as even though HTTPS traffic was being captured, there was still unreadable data being sent to servers that there wasn't any documentation on.

Apart from this, it would be interesting to compare the data here with data from browsers on Android/iOS, to see how much more or less diagnostic data are collected between platforms.

References

- Google. (2020). *Chrome Variations*. Retrieved from <https://www.google.com/chrome/privacy/whitepaper.html#variations>
- Google. (2020). *GAIA Access Control*. Retrieved from <https://developers.google.com/issue-tracker/concepts/access-control>
- Mozilla. (2020). *Firefox continues push to bring DNS over HTTPS by default for US users*. Retrieved from <https://blog.mozilla.org/blog/2020/02/25/firefox-continues-push-to-bring-dns-over-https-by-default-for-us-users/>
- Mozilla. (n.d.). *Firefox Public Data Report*. Retrieved from <https://data.firefox.com/dashboard/user-activity>
- Mozilla. (n.d.). *What is the Mozilla Maintenance Service?* Retrieved from Mozilla: <https://support.mozilla.org/en-US/kb/what-mozilla-maintenance-service>
- W3Counter. (2020, April). *Browser & Platform Market Share*. Retrieved from <https://www.w3counter.com/globalstats.php>