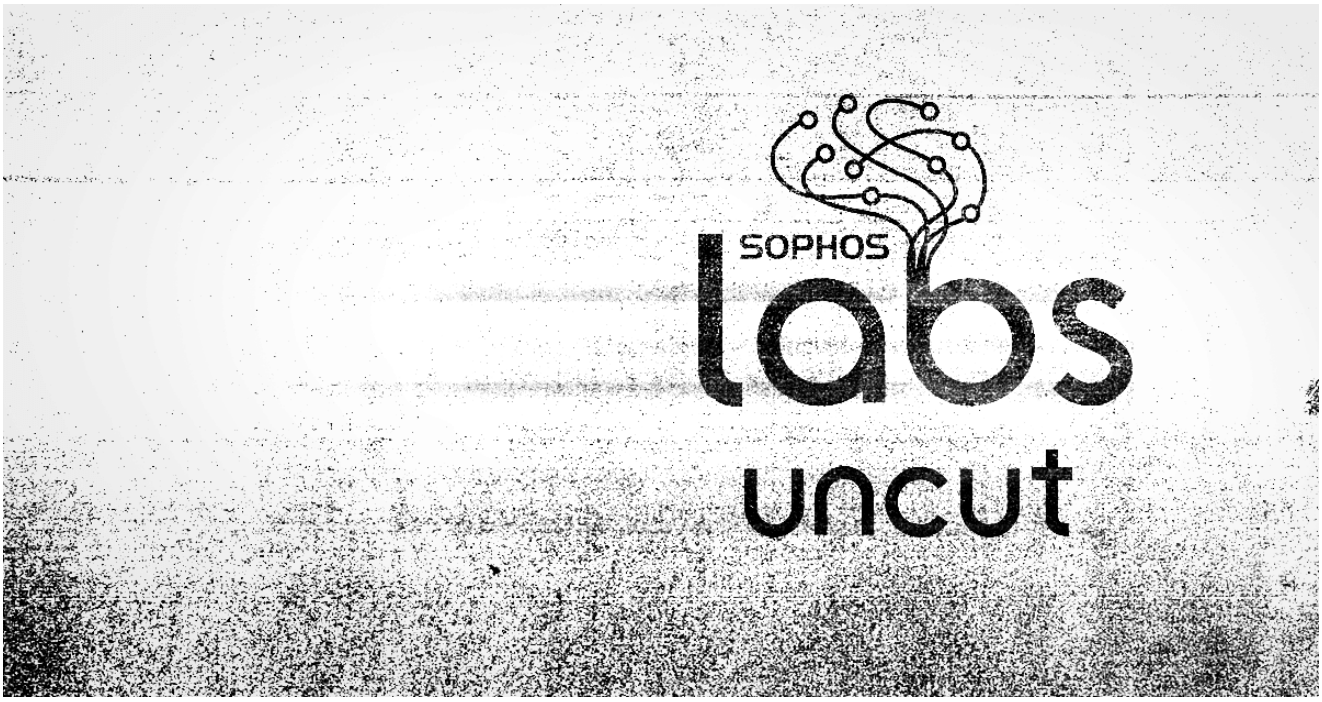
AdKoob information thief targets Facebook ad purchase info

news.sophos.com/en-us/2018/07/29/adkoob-information-thief-targets-facebook-ad-purchase-info/

Felix Weyne July 29, 2018



By Felix Weyne

At Sophos, we are continuously on the lookout for new threats. One of the systems which helps us in sifting through the daily volume of fresh malware is our sandbox environment, which gives us the ability to analyze the malware's dynamic (runtime) behaviour.

Recently, we identified a suspicious executable which showed intriguing behaviour in our sandbox. The executable injected code into a legitimate windows binary (svchost.exe), and the injected code triggered one of our memory detections which aims to identify *information stealing malware*. The injected process ended abruptly and displayed an error message which didn't make any sense relative to the type of code which the process contained: "The configuration file is missing. Re-installing Easy Backup may fix this problem".

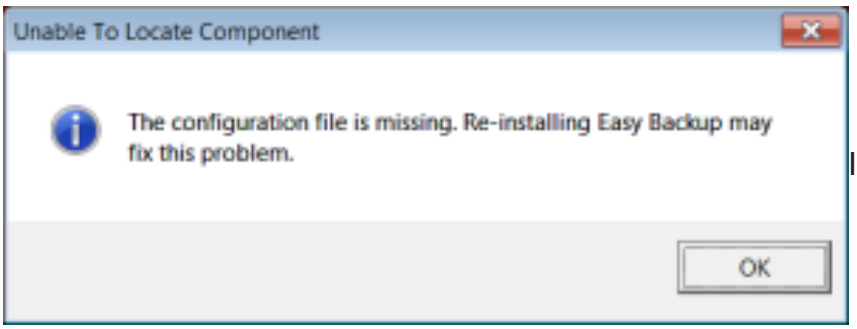


Image one: Decoy messagebox displayed by AdKoob

This odd behaviour prompted us to analyze the malware in depth. We took a jump down the rabbit hole, leading to the discovery of a previously undocumented threat, which we're calling **AdKoob**. It's a credential stealing malware with a twist: It also exfiltrates data from your Facebook account to an unknown recipient.

Evading detection and sandbox

In its effort to avoid being detected, the main AdKoob executable is double-packed: First, with the open source UPX packer, and then with a custom injector. The custom injector uses a technique called process hollowing to map the malware code in a suspended instance of the Windows Service Host executable. Once mapped, the injector resumes the Service Host process, at which point the malicious code runs, and the fake error message pops up.

But when we disassembled AdKoob, we quickly figured out why it throws a fake error message: AdKoob starts its execution by checking if there is a command line argument present. If there isn't one, it throws an error, but in one sample I analyzed, it looks for an argument of either `'/1c9542b2a8cb'` or `'/mode=debug'`.

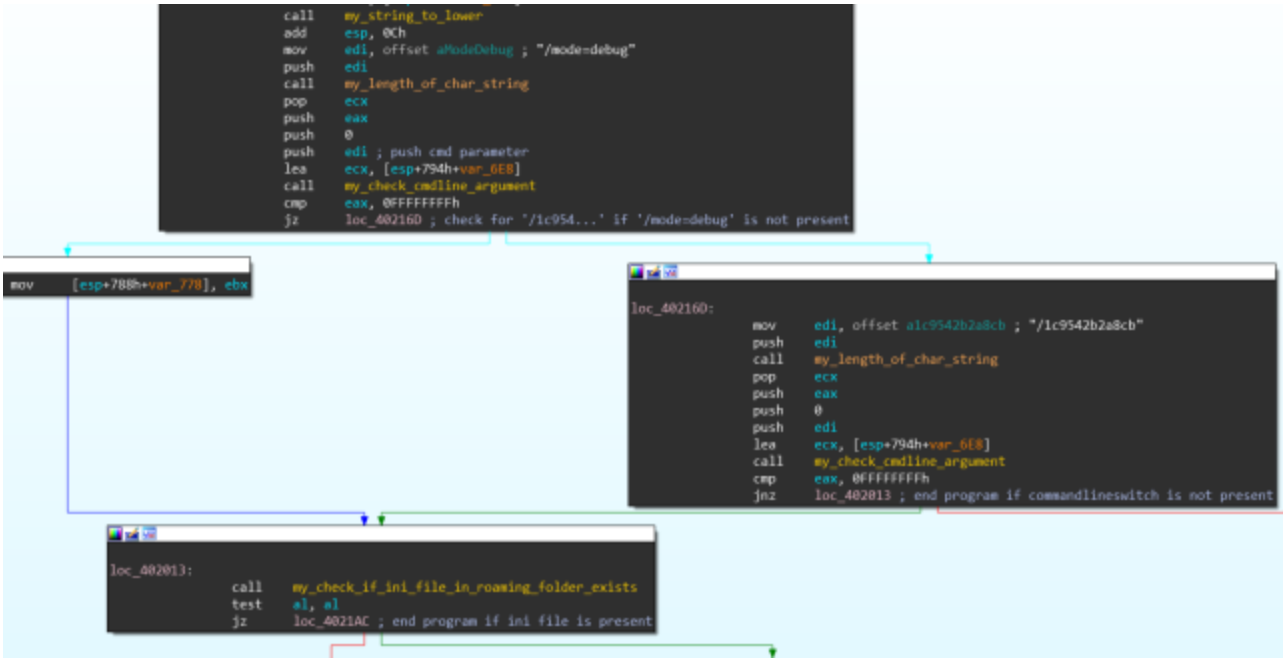


Image two: Disassembled start of AdKoob

If whatever process starts the malware passes it the correct argument, AdKoob continues its execution by checking for the presence of `'FC29FA0894FE.ini'` in `%appdata%`, which functions as a breadcrumb to ensure the malware only gets executed once. If the breadcrumb file exists, the malware terminates immediately.

Using a built-in command line parameter check is a simple yet crafty anti-sandbox technique, since it is unlikely that an automated sandbox will know in advance which parameter it needs to pass to the executable in order to trick it into revealing its true colours.

Stealing browser credentials

Once all the evasion checks are satisfied, AdKoob starts its first core task: stealing usernames and passwords saved in the browser. To achieve this goal, AdKoob directly accesses the file and registry locations where various browsers store the credentials.

AdKoob has to use a few tricks in order to steal data from different browsers. For Chrome and older versions of Firefox (prior to version 58), it does a SQL query against the SQLite database that these browsers store on disk (`moz_logins` for Firefox, and `logins` for Chrome).

```
SELECT encryptedUsername, encryptedPassword, formSubmitURL FROM moz_logins
SELECT origin_url, username_value, password_value FROM logins
```

AdKoob accesses the SQLite databases by filename:

```
signons.sqlite (Firefox's SQL credential database)
Google\Chrome\User Data\Default\Login Data (path to Chrome's database)
```

Recent versions of Firefox store credentials within a JSON file, which AdKoob is also able to query.

Additionally AdKoob looks for saved login form credentials of Internet Explorer in the following registry location:

```
Software\Microsoft\Internet Explorer\IntelliForms\Storage2
```

AdKoob is able to extract Internet Explorer's saved 'Basic Access Authentication' credentials (these are used by certain web applications, sent within specific HTTP headers). It uses the GUID string required to generate the cryptographic salt that protects the saved credentials, and the prefix needed to distinguish the saved basic access credentials from other credentials in the vault.

```
abe2869f-9b47-4cd9-a358-c22904dba7f7 (GUID string used as cryptographic salt)
Microsoft_WinInet (Prefix to identify stored Basic access authentication credentials)
```

Exfiltrating harvested credentials

After harvesting the credentials, AdKoob determines the victim's public IP address by making a HTTP request to either `'useragent.cc'` or `'mybrowserinfo.com'`. Besides the victim's public IP address, basic information about the machine is also collected.

Harvested browser credentials are encoded and then sent to the attacker over HTTPS, using two URLs hardcoded in the malware. A separate POST request is made for each browser. For the analyzed sample, the exfiltration URLs (obfuscated here) were:

```
hxxps://104[.]200[.]131[.]253:1989/stats1.asp
hxxps://45[.]32[.]91[.]128:1989/stats1.asp
```

An example decoded request looks like:

```
3ED4ACD2B09E4389E997B918A9A7ADB4B07A4611BF|Windows NT 6.1<<1.2.3.4|UTC+00:00 GMT
Standard Time|chrome|Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/67.0.3396.99
Safari/537.36{||}aHR0cHM6Ly9hY2NvdW50cy5nb29nbGUuY29tL3NpZ25pbi92Mi9zbC9wd2Q=|dGVzdGNy
```

The fields in the decoded request are described below:

Field	Data
Unique machine identifier (derived from hostname & volume serial number of the C-drive)	3ED4ACD2B09E4389E997B918A9A7ADB4B07A4611BF
OS version	Windows NT 6.1
Public IP (example)	1.2.3.4
Victim's local timezone	UTC+00:00 GMT Standard Time
AdKoob's browser identifier string	chrome
User agent (of the browser from which the credentials were stolen)	Mozilla/5.0 (Windows NT 6.1; Win64; x64)…
Base64 encoded browser credential(s) (website|username|password)	aHR0cHM6Ly9…|dGVzdGNyZWQ=|U2F2ZU1lIQ==
Bot identifier	yangyangfb

Facebook profile espionage

Once the saved browser credentials are exfiltrated from the victim's machine, AdKoob starts its second core task: establishing a Facebook session and exfiltrating data from the victim's Facebook profile. AdKoob uses two methods to establish a Facebook session.

The first method relies on using an existing Facebook session stored in the user's browser via Facebook authentication cookies. AdKoob goes through the victim's installed browsers, and specifically looks for cookies related to the facebook.com domain.

```
.rdata:004C60C0 aSGoogleChromeU:                        ; DATA XREF: my_steal_facebook_credentials_in_browser+E2↑o
.rdata:004C60C0 text "UTF-16LE", '%s\Google\Chrome\User Data\Default',0
.rdata:004C6106 align 4
.rdata:004C6108 ; char aHostKey[]
.rdata:004C6108 aHostKey db 'host_key',0                 ; DATA XREF: sub_41C51C+A5↑o
.rdata:004C6111 align 4
.rdata:004C6114 ; char aEncryptedValue[]
.rdata:004C6114 aEncryptedValue db 'encrypted_value',0  ; DATA XREF: sub_41C51C+11A↑o
.rdata:004C6124 aS_2 db '%S',0                          ; DATA XREF: sub_41C51C+14A↑o
.rdata:004C6127 align 4
.rdata:004C6128 ; char aExpiresUtc[]
.rdata:004C6128 aExpiresUtc db 'expires_utc',0          ; DATA XREF: sub_41C51C+179↑o
.rdata:004C6134 align 8
.rdata:004C6138 aSelectFromCook db 'SELECT * FROM cookies WHERE cookies.host_key LIKE "%.facebook.com'
.rdata:004C6138                                         ; DATA XREF: my_steal_fb_cookie_chrome+7E↑o
.rdata:004C6138 db '";',0
.rdata:004C617C aCookies:                               ; DATA XREF: my_steal_fb_cookie_chrome:loc_41C7BD↑o
.rdata:004C617C text "UTF-16LE", 'Cookies',0
```

Image Three: AdKoob looking for Facebook cookies stored in Chrome browser

Similar to saved credentials, cookies are stored in a SQLite database in Firefox and Chrome. We can easily spot the following SQL queries used by AdKoob to query the cookies:

```
SELECT * FROM cookies WHERE cookies.host_key LIKE "%.facebook.com";
SELECT * FROM moz_cookies WHERE moz_cookies.host LIKE "%.facebook.com";
```

Besides Chrome and Firefox, AdKoob also queries the cookies of Internet Explorer and Microsoft Edge for the presence of Facebook cookies.

The second method involves using the browser credentials AdKoob has already stolen. If any of these are for Facebook, AdKoob attempts to log in to Facebook by sending the appropriate HTTP requests. If successful, it is able to get hold of additional Facebook session cookies.

Once successfully authenticated to Facebook, AdKoob proceeds with its espionage payload. It launches a series of requests to a list of Facebook URLs. The malware inspects the response data from each request using a corresponding set of regular expressions. This technique enables AdKoob to identify items of interest within the victim's Facebook profile.

The first three requests are to the following Facebook URLs:

```
https://www.facebook.com/
https://www.facebook.com/username/about
https://www.facebook.com/bookmarks/pages
```

From these requests, AdKoob exfiltrates the following data:

1.
    - Facebook Locale (parameter which defines the user's preferred interface language, region)
    - Facebook User ID (a string of numbers that doesn't personally identify a Facebook user but does connect to a Facebook user's profile)
    - Facebook Profile username (username chosen by the user, which is used in the custom link to the users's Facebook profile, e.g. www[.]facebook.com/janedoe3)
    - Various info from the about section of the user's Facebook profile (including the user's full name and birthday).
    - Name(s) of Facebook page(s) the user created

Interest in advertisement spending

After having spied on the victim's Facebook profile data, AdKoob continues by making a series of interesting requests to an interface which isn't used by everyday Facebook users, relating to paid advertisement campaigns. Businesses have the possibility to create an advertisement account on Facebook, and use that account to promote a Facebook page or website via a specific campaign. Various parameters can be defined by the user, for example the demography of users on which the campaign should focus.

For each advertisement campaign, the user can define a budget via the Facebook Ads manager interface, where he can also get a clear overview on the results and spending of the advertisement campaign.

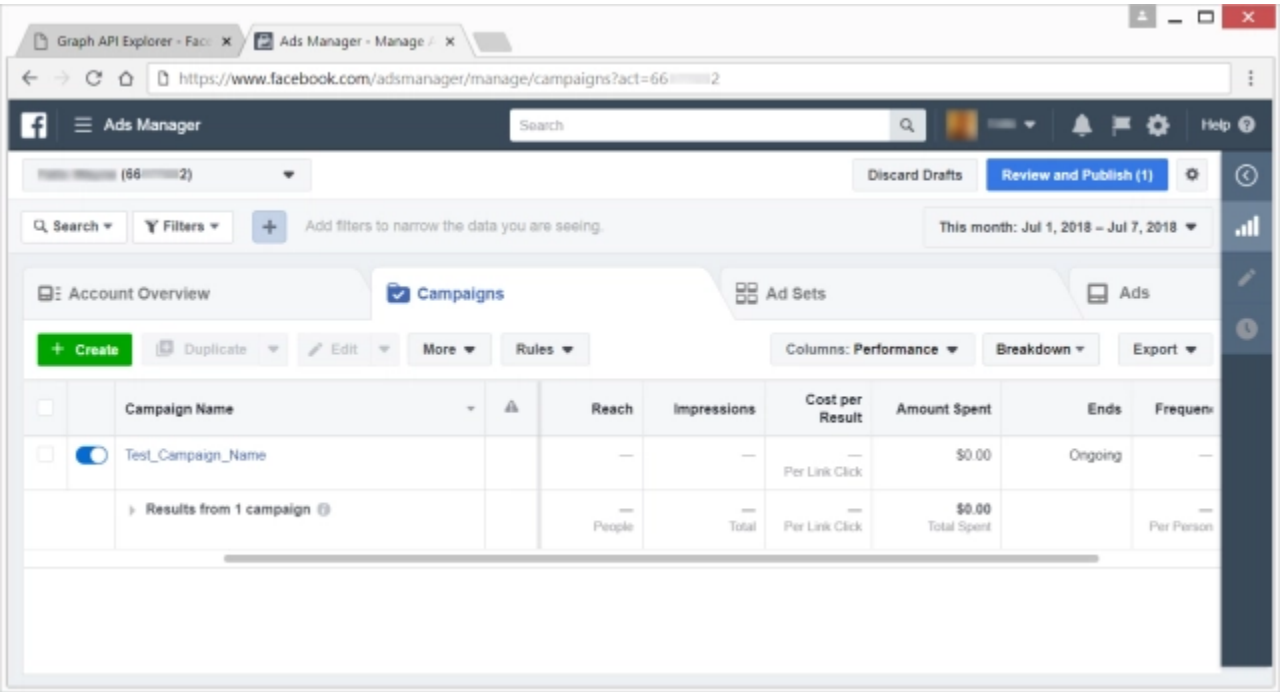


Image four: Facebook's ads manager interface

AdKoob makes a request to the URL on which this interface is served. It then checks the response and extracts the ad account ID (a string of numbers which uniquely identify the advertisement account) and the total amount of money which has been spent on

advertisement campaigns.

The extracted ad account ID is then used to make a query to the Facebook Graph API, a programmer's interface which allows a programmer to query information from a Facebook account. The following query is launched by AdKoob:

```
act_<adaccountid>/campaigns?date_preset=this_month&fields=
["boosted_action_type","brand_lift_studies","buying_type","effective_status","name","s
[]&include_headers=false&limit=20&locale=en_US&method=get&pretty=0&sort=
["delivery_info_ascending","spent_descending","stop_time_descending","start_time_desce
["total_count"]&suppress_http_code=1
```

The query and corresponding result can be simulated via the graphical Facebook graph API explorer, a Facebook tool for developers to test their Facebook graph queries, as shown in image five.

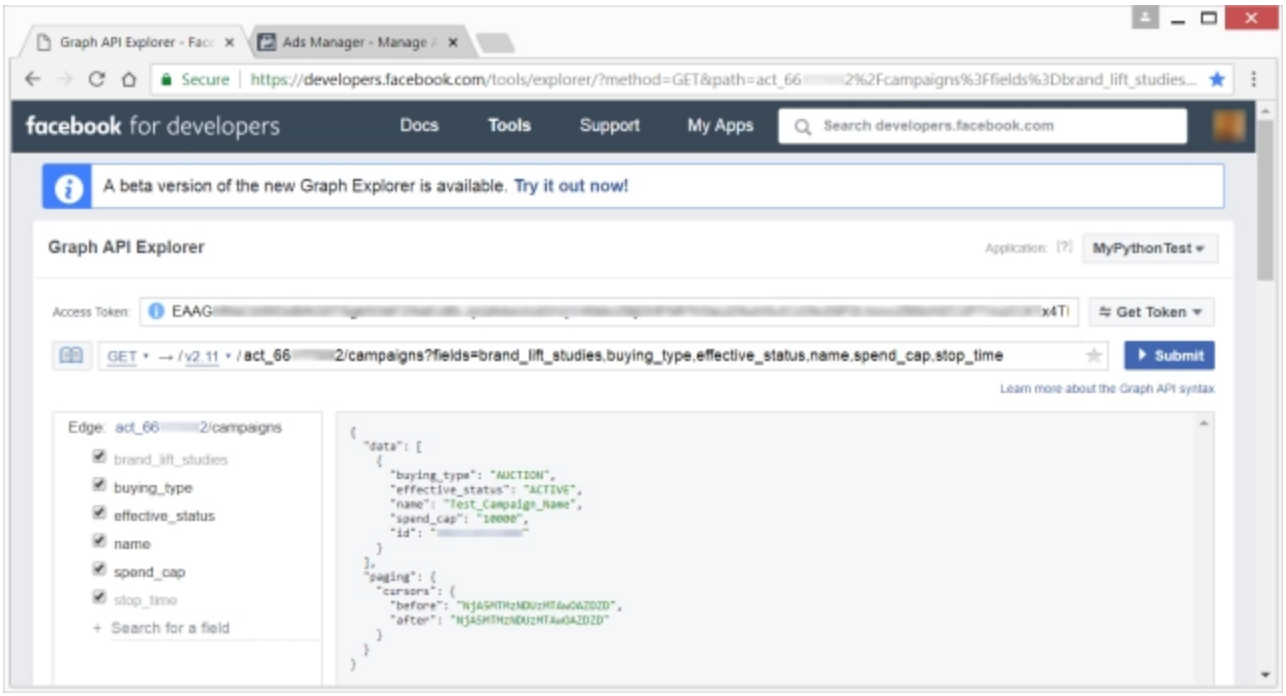


Image five: AdKoob's query emulated in Facebook's Graph API Explorer

The query launched by AdKoob steals various information from advertisement campaigns, such as its campaign name, amount of money currently spent and the ad campaign's budget limit.

This is perhaps the most interesting thing about AdKoob – the payload suggests the criminals behind the attack are targeting business users. For the simple reason that regular home users are unlikely to use Facebook advertisement campaigns.

All the extracted Facebook data gets sent to the same hardcoded URLs as those which were used to exfiltrate the browser credentials. In addition to the stolen data from the Facebook account, the Facebook cookies and the Facebook login credentials (if present on the victim's machine) are also exfiltrated.

Once all exfiltration is complete, AdKoob sends a GET request to a different URL, which includes an identifier based on the user's Security Identifier (SID):

```
https://107[.]151[.]152[.]220:5658/down.asp?action=newinstall&u=<identifier>
```

This URL is different from the URLs used for data exfiltration, and presumably is used to gather statistics about the number of infections.

As a finale, AdKoob ends its execution by creating the file `'%appdata%\FC29FA0894FE.ini'`, which ensures the malware isn't run twice. It then self-destructs in an attempt to leave no sign that the system was compromised.

Sophos detects this threat as Troj/AdKoob-A.

IOCs

Packed AdKoob sample (SHA256):
e383582413cc53ec6a630e537eedfeee35d6b5f3495266f2530770f4dd3193a6
Unpacked, analyzed AdKoob sample (SHA256):
6a6260cb5e1e0ad22af2bc8bb73bc8423df6315a88e39c2264f6def5798b6550

YARA Rule:

```
rule adkoob_information_stealer
{
    meta:
        author = "Felix Weyne, Sophos"
    strings:
        $facebook_cookie_firefox = "SELECT * FROM moz_cookies WHERE moz_cookies.host
LIKE \"%.facebook.com\""  nocase ascii
        $facebook_cookie_chrome = "SELECT * FROM cookies WHERE cookies.host_key LIKE
\"%.facebook.com\""  nocase ascii
        $facebook_regex_ad_account_id = "<td [^>]*?data-
testid=\"all_accounts_table_account_id_cell\">([^<>]*?)</td>"  nocase wide
        $self_destruction = "/C ping localhost -n 4 > nul & del" nocase wide
    condition:
        all of them
}
```