VB2021
localhost

7 - 8 October, 2021 / vblocalhost.com

HUNTING WEB SKIMMERS WITH VIRUSTOTAL AND YARA

Jérôme Segura
Malwarebytes, Canada

jsegura@malwarebytes.com

ABSTRACT

Does shopping online sometimes feel like playing Russian roulette? During the past few years, the web threat landscape has seen an increase in JavaScript-based credit card skimmers also known under the name Magecart. These code snippets can steal your payment information and other personal details in the blink of an eye.

To keep up with this threat we can deploy various tools such as web crawlers that mimic a user browsing to an online store in order to collect any malicious code loaded during the process.

We introduce an additional tool that is less infrastructure-heavy since it relies on using a combination of *VirusTotal* and YARA rules. We can automate the process of extraction of newly compromised online shops as well as skimmer gates used by criminals for data exfiltration.

In this paper we will look at:

1. The basics of web skimmers: understanding how malicious JavaScript captures and exfiltrates data in real time.
2. The current skimmer detections from anti-virus engines: several vendors can detect Magecart injections inside static files.
3. Building YARA rules to detect web skimmers: creating your own YARA signatures to expand your searches and catch new skimmer families.
4. Identifying skimmer gates and victim sites: from HTML and JavaScript source files you can find out which sites are hacked and which are used as infrastructure.
5. Automating web skimmer detection: a Python script that lets you automate collection and processes results.

We hope to provide those interested in web skimming with one more weapon in their arsenal to start hunting down and reporting on new Magecart attacks.

INTRODUCTION

The web threat landscape has evolved a fair bit during the past few years; in particular one of the biggest changes has been the decline of exploit kits primarily targeting *Internet Explorer* as the market share for the legacy *Microsoft* browser has become much smaller.

Meanwhile, threat actors have actively been targeting *Chromium*-based browsers in a variety of malvertising-driven campaigns with social engineering components. But infecting users with malware is not the only driver at play, and sometimes it is just easier and more financially profitable to steal from them without compromising their devices at all.

Online shopping is a billion-dollar industry that grew by 44% in the US in 2020 [1]. Practically all big and even small brands have an online store and have seen online retail become a bigger part of their revenues. These new realities and behaviours have attracted criminals looking to target customers directly in ways that weren't possible, or perhaps not as common, before.

Enter web skimmers, the digital equivalent of ATM skimming targeting online stores. This threat is known under many different names including sniffers, digital skimmers, web skimmers, and of course Magecart. The latter name was coined by *RiskIQ* and is a play on *Magento*, a popular content management system and online shopping cart. Although skimmers have been around for many years, it wasn't until major incidents targeting *British Airways* [2] and *Ticketmaster* [3] happened that Magecart reached the mainstream [4].

Probably one of the best ways to track skimmers at scale is to use web crawlers and some automation that includes simulated interaction. While some companies can afford to deploy and maintain this kind of infrastructure, many researchers can't.

Although the popular malware repository and hunting platform *VirusTotal* is primarily known for analysing PE binaries, its data corpus is much larger and includes HTML and JavaScript files. Those with a *VirusTotal* licence can hunt for Magecart detections from a number of anti-virus products as well as leverage the power of YARA to create their own rules and detect new skimming infrastructure and victim sites.

WEB SKIMMERS BASICS

When a customer buys a product or service online, they usually need to provide information including the most interesting one for criminals: their credit card number. This data is entered into a payment form before it is validated and processed.

Criminals don't need to infect each and every user with malware in order to collect this data; they can simply do it at scale by inserting themselves into the online payment flow of dozens or hundreds of stores at once. There are a number of ways this can be done but they all have the same end goal.

Popular e-commerce platforms such as *Magento* are, just like other content management systems, vulnerable to security flaws and need to be patched regularly. Criminals can compromise them [5] and inject malicious code designed to monitor and steal keystrokes happening within the browser page.

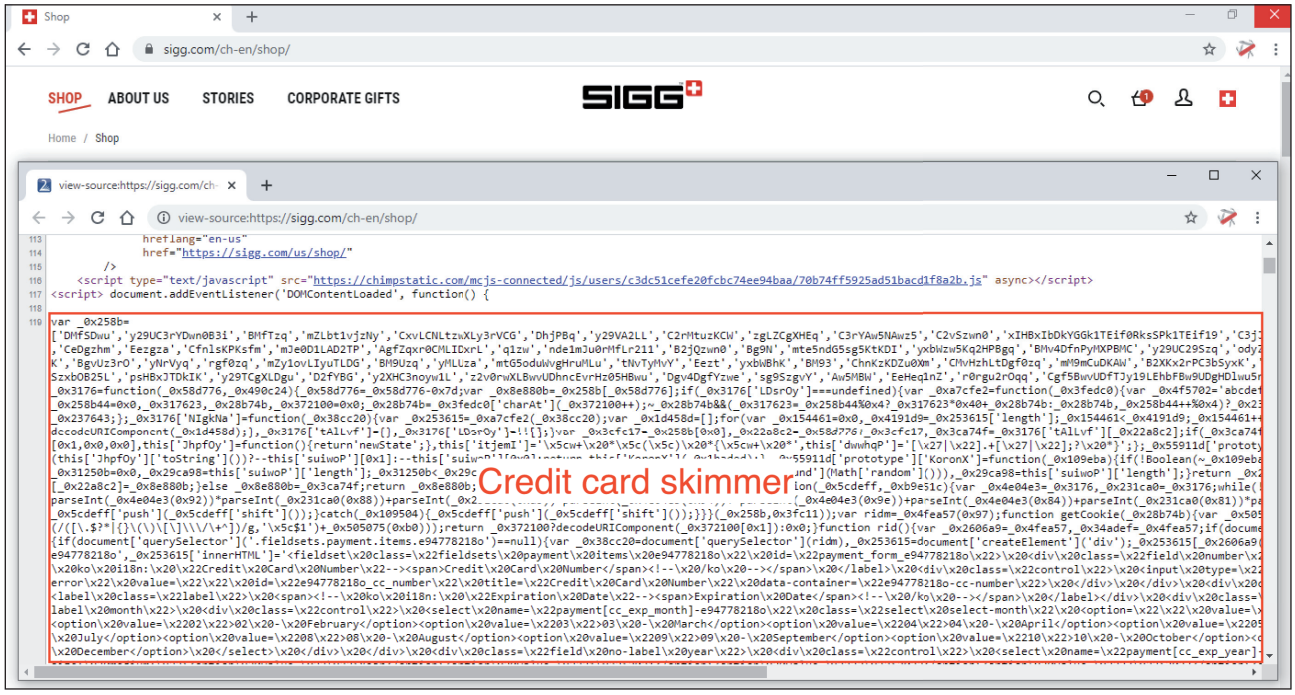


Figure 1: An online store that has been injected with an obfuscated skimmer.

Most skimming code is obfuscated JavaScript that hides its actual intentions but generally provides the following functionality:

- Check if the current page is the checkout
- Check if developer tools or other debugging artifacts are present
- Monitor payment form fields at regular intervals
- Exfiltrate captured data to a criminal-owned server (gate).

Essentially, a skimmer will ensure the victim is at a checkout page, steal their data as they type it in, and then exfiltrate it. Contrary to popular belief, it doesn't matter if the shopping site is using HTTPS or not: the data is only protected when in transit between the browser and merchant site. A skimmer steals that data when it is 'at rest' and being entered into form fields.

As with other types of website malware, skimmers can usually be recognized by specific patterns. In fact, this is the basis for signature-based detection and can help to categorize skimmers into various sub-families. In the next sections of this paper we will dig deeper into how these patterns can be used to identify new compromises, both by leveraging anti-virus signatures and creating your own.

ANTI-VIRUS DETECTIONS

Even though anti-virus products primarily focus on traditional malware (i.e. PE files), a number of them can scan and detect malicious code hidden in HTML or JavaScript. This can be a good starting point to start looking at popular web-skimming code that can be identified via AV signatures.

VirusTotal Intelligence allows you to perform search queries that look for specific detection names. Just as in the malware world, security vendors don't seem to agree on any naming convention, so you will likely need to create your own dictionary list of keywords (Magecart, Skimmer, CardStealer, MagentoStealer). A basic search query [6] looking for those anti-virus labels returns over 47,000 results.

We can also build YARA rules that look for those strings within the metadata collected from a *VirusTotal* scan. To make the rule more specific to web skimmers, we can also filter the file types to scripts and HTML source code. We can use a YARA rule for *Livehunt* [7] and add the additional filter field of *vt.metadata.new_file* to ensure the sample was submitted to *VirusTotal* for the first time.

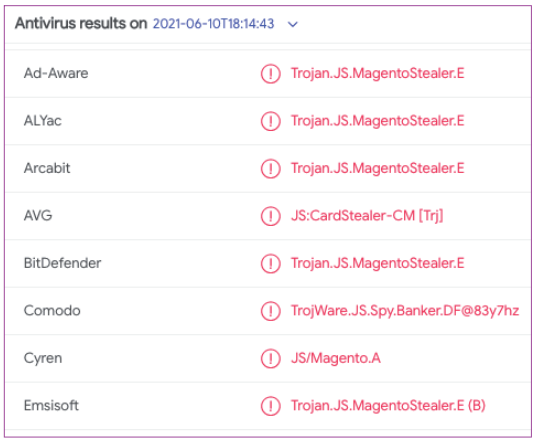


Figure 2: Anti-virus detections for a web skimmer.

```
import "vt"
rule AVsigs_WebSkimmer : Magecart WebSkimmer {
    meta:
       author = "Jérôme Segura"
       description = "Skimmer rule from AV detections"
       reference = "https://github.com/malwareinfosec/webskimmers"
       date = "2021-06-03"
    condition:
       for any engine, signature in vt.metadata.signatures : (
           (signature contains "Magecart" or signature contains "Skimmer" or signature contains
"CardStealer" or signature contains "MagentoStealer")
       ) and (vt.metadata.file_type == vt.FileType.SCRIPT or vt.metadata.file_type ==
vt.FileType.HTML) and vt.metadata.new_file
}
```

Reviewing notifications and downloading them locally is a really good exercise. It allows you to study each sample and figure out where skimming code is located. This, in turn, will enable you to begin writing your own rules, without depending on AV engines.

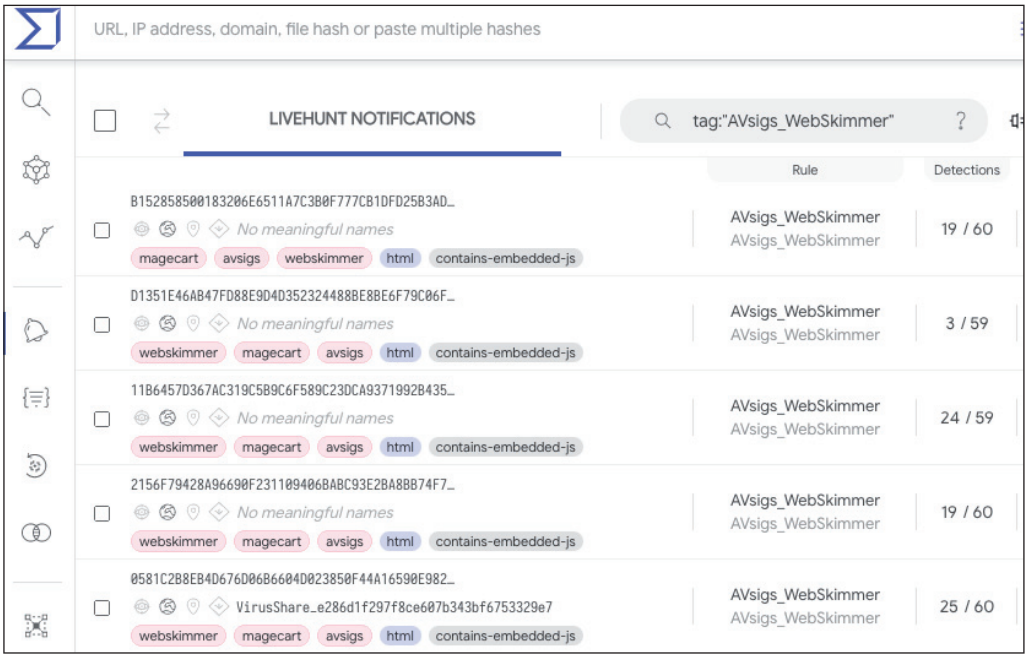


Figure 3: Livehunt notifications.

FINDING UNDETECTED SKIMMERS

The next step in your hunt for web skimmers is to start building your own rules. The idea is that AV detections alone are not complete and that leveraging the power of YARA allows you to explore wider rules that may generate some false positives but could also find new variants.

There are a number of different sources to help with collecting skimmers, ranging from security blog posts to following the Magecart hashtag on *Twitter* [8]. Sites such as *urlscan.io* [9] also provide saved HTML and JavaScript responses that can be downloaded.

```
import "vt"
rule simple_WebSkimmer : Magecart WebSkimmer
{
    meta:
       description = "Simple Skimmer"
       reference = "https://twitter.com/AffableKraut/status/1399786791931101192"
    strings:
       $re1 = /=\s\["change",\s"\[name=cc_cvv2\]",/
       $s1 = "post"
       $s2 = "ready"
    condition:
       all of them and (vt.metadata.file_type == vt.FileType.SCRIPT or vt.metadata.file_type ==
vt.FileType.HTML)
       and vt.metadata.new_file
}
```

Many skimmers will contain unique text strings that can be used 'as is' to create a rule. For example, the 'Grelos' skimmer [10] was named after the variable '*var grelos_v*'. Naming skimmers after such strings is a fairly common practice, although it does not take into account which threat group might be behind it.

Because skimmers revolve around stealing credit card data, rules can be written around specific fields such as 'cvv', 'expiry date', etc. Often, there are unique ways in which the malware authors will grab that information both in clear text (i.e. 'GetCCInfo') and obfuscated or encoded formats (i.e. 'X2NjX2V4cF9tb250aA==' for '_cc_exp_month').

Heuristic rules may look for certain artifacts that accompany skimmers, such as the detection of browser developer tools ('*devtools.open*'). This kind of check is quite common and ensures that skimming code is not being debugged by a security researcher. Because a number of legitimate websites also perform that check, it is best used in conjunction with other search terms, for example, combining Devtools with 'checkout|onepage' to narrow down the search to e-commerce sites.

Malware authors love to play with file formats. We have seen the same done with web skimmers leveraging steganography to hide inside image files [11]. There are different ways to identify images containing extra data such as looking at their markers, in particular the end of file marker for specific formats like JPEG or PNG. It's also worth checking the EXIF metadata for JavaScript code within certain fields such as Exif.Image.Copyright [12].

All those YARA rules can be added to your *Livehunt* ruleset and *VirusTotal* will send notifications when it identifies a match.

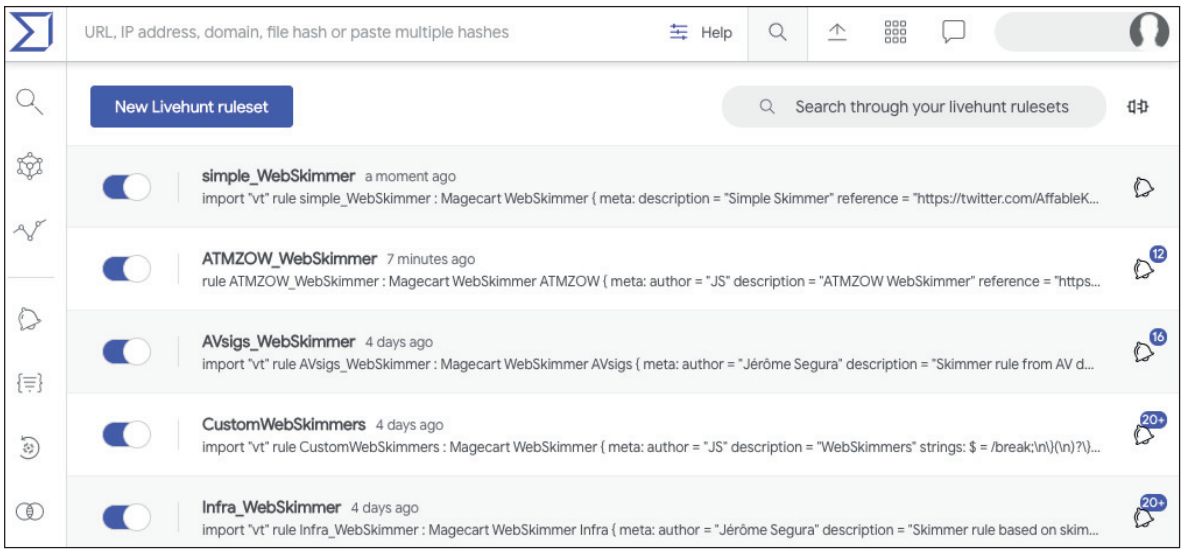


Figure 4: VirusTotal Livehunt YARA ruleset.

EXTRACTING SKIMMER INFRASTRUCTURE AND UNCOVERING NEW VICTIM SITES

The process of extracting skimmer infrastructure is probably the most difficult because it often involves decoding JavaScript that may have been well obfuscated. What we typically look for is known as a 'gate', which is where the threat actors will collect stolen credit card data sent by the skimmer itself.

Sometimes the gate will be visible in plain text within the code (*Gate: "https://jquerycdnlib.at/gate.php"*,) or it might be Base64 encoded. In both cases, it is fairly easy and quick to extract it.

Collecting such data is particularly important for defenders. For one, any malicious domain can be added to a blocklist as part of the web protection component of a security product. This ensures that your users will be protected against known infrastructure hosting skimmer code or exfiltrated data. It's also a good starting point for a new investigation that can uncover additional domains or IP addresses that may ultimately tie back to a known threat group. Platforms such as *RiskIQ*'s *PassiveTotal* [13] allow this kind pivoting and attribution.

```
# Find skimmer gate from file
def find_gate(sha256, data_tmp):
    skimmer_gate = None
    try:
        with open(data_tmp + sha256, 'r', errors='replace') as f:
            for line in f.readlines():
                if 'Gate: "' in line:
                    skimmer_gate = re.sub('",', '', re.sub('^.*?Gate:\s'https?://(www.)?', '', line)).strip('\n')
                    if skimmer_gate is not None:
                        return skimmer_gate
                if 'x68' in line:
                    hex_tmp = re.search('\\\\x68\\\\x74\\\\x74\\\\x70\\\\x73\\\\x3A\\\\x2F\\\\x2F(.*?)\\\\x2F', line, re.IGNORECASE)
                    if hex_tmp is not None:
                        hex_tmp = hex_tmp.group(1)
                        skimmer_gate = hex_text(hex_tmp)
                        return skimmer_gate
```

Figure 5: Python code to find skimmer gates.

Finally, newly identified skimming infrastructure can be fed into YARA rules to uncover additional victim sites. One of the most common website infection patterns is to load a third-party script from a remote website. Your YARA rule will look for the presence of known malicious domains in the HTML source code of e-commerce sites. To build this rule from a list of domain names, you can use the script at [14].

Figure 6: YARA rule to identify victim sites from known malicious infrastructure.

Finding victim sites is much easier in comparison, provided you have the HTML source code available instead of a JavaScript library, which could be loaded from just about anywhere.

While not always a productive effort, reporting compromised websites raises awareness about the threat of web skimmers and web security in general. Many business owners believe that their platform is safe out of the box, especially if they rely on external payment processors. But criminals are smart at finding ways to phish users [15], even when a site would not directly accept payments.

```python
# Find victim site from file
def find_victim(sha256, data_tmp):
    victim_site = None
    try:
        with open(data_tmp + sha256, 'r', errors='replace') as f:
            for line in f.readlines():
                if ('link rel="shortcut icon' in line) and (line.find('http') != -1):
                    victim_site = re.sub('/.*', '',re.sub('^.*shortcut icon.*?href="https?://(www.)?', '', line)).strip('\n')
                    return victim_site
                elif ('var BLANK_URL = ' in line) and (line.find('http') != -1):
                    victim_site = re.sub('/.*', '', re.sub('^.*?(\'|")https?://(www.)?', '', line)).strip('\n')
                    return victim_site
                elif ('<base href="http' in line):
                    victim_site = re.sub('/.*', '', re.sub('^.*?(\'|")https?://(www.)?', '', line)).strip('\n')
                    return victim_site
                elif ('<link rel="pingback"' in line) and (line.find('http') != -1):
                    victim_site = re.sub('/.*', '', re.sub('^.*?(\'|")https?://(www.)?', '', line)).strip('\n')
                    return victim_site
                elif ('"baseUrl": "' in line) and (line.find('http') != -1):
                    victim_site = re.sub('/.*', '', re.sub('^.*?(\'|")https?://(www.)?', '', line)).strip('\n')
                    return victim_site
    except Exception as e:
        with open(missing_log_file, 'a') as f:
            f.write("[!] Error with finding victim_site with: " + sha256 + " : " + str(e))
```

Figure 7: Python code snippet to find new victim sites.

AUTOMATING WEB SKIMMER DETECTION

While you can perform manual hunting and discover interesting data, eventually you may want to start automating certain things. Thanks to the *VirusTotal* APIs we can query the notifications for our YARA rules used in *Livehunt* mode.

Leveraging our knowledge on uncovering infrastructure and victim sites allows us to collect additional information from any file that was uploaded to *VirusTotal* and matched one of our rules.

The author is providing a Python script [16] with the following functionality:

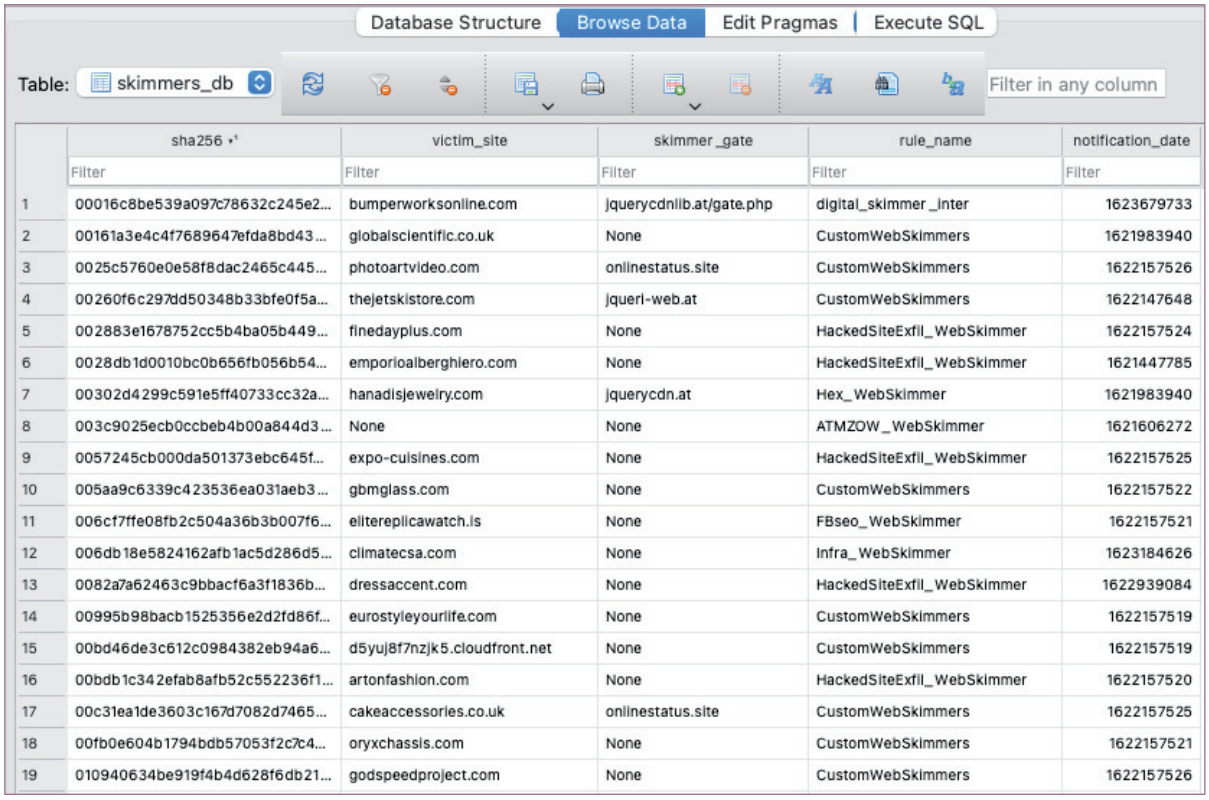
- Query *VirusTotal Livehunt* notifications
- Download matches locally
- Display which YARA rule matched
- Extract the victim site (if applicable)
- Extract the skimmer gate (if possible)
- Store the matched file's SHA256, matching rule, victim site and gate into a local database

```
--------------------------------------------------------------------------------
Rule name: Inter_WebSkimmer
Match date: 06/14/2021
SHA256: d65709dee588becbde06acba495dc066cb1613e8a012e84fcb8fba3d9c90098a
Victim site: marineoutlet[.]com[.]br
New victim site: No
Skimmer gate: jquery-script[.]icu/gate
New skimmer gate: No
--------------------------------------------------------------------------------
Rule name: HackedSiteExfil_WebSkimmer
Match date: 06/14/2021
SHA256: 4c0636415459d3836d7cf83361c3ee5d5ace6979e0167c9472889da4036fbf6f
--------------------------------------------------------------------------------
Rule name: HackedSiteExfil_WebSkimmer
Match date: 06/14/2021
SHA256: c403370cafd0ef8bc967f79316bd6b3f96584798b3d5b9cc50ab468a7bd3eaa4
Victim site: marqqa[.]com
New victim site: Yes
--------------------------------------------------------------------------------
Rule name: Stegano_WebSkimmer
Match date: 06/14/2021
SHA256: 8cf2e5547e5343e9a32e71a38dbb14c34413b45d130370986faa679121c15141
--------------------------------------------------------------------------------

STATS:
Victim sites found: 6
Victim sites missed: 2
Skimmer gates found: 1
Skimmer gates missed: 7
```

Figure 8: Output from script that identifies web skimmers.

The script can be run on demand or at regular intervals and returns a list of matches for specific SHA256 hashes. The results are stored in a local SQLite database which allows new notifications to be checked against this database.



Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: skimmers_db — Filter in any column

	sha256	victim_site	skimmer_gate	rule_name	notification_date
	Filter	Filter	Filter	Filter	Filter
1	00016c8be539a097c78632c245e2...	bumperworksonline.com	jquerycdnlib.at/gate.php	digital_skimmer_inter	1623679733
2	00161a3e4c4f7689647efda8bd43...	globalscientific.co.uk	None	CustomWebSkimmers	1621983940
3	0025c5760e0e58f8dac2465c445...	photoartvideo.com	onlinestatus.site	CustomWebSkimmers	1622157526
4	00260f6c297dd50348b33bfe0f5a...	thejetskistore.com	jqueri-web.at	CustomWebSkimmers	1622147648
5	002883e1678752cc5b4ba05b449...	finedayplus.com	None	HackedSiteExfil_WebSkimmer	1622157524
6	0028db1d0010bc0b656fb056b54...	emporioalberghiero.com	None	HackedSiteExfil_WebSkimmer	1621447785
7	00302d4299c591e5ff40733cc32a...	hanadisjewelry.com	jquerycdn.at	Hex_WebSkimmer	1621983940
8	003c9025ecb0ccbeb4b00a844d3...	None	None	ATMZOW_WebSkimmer	1621606272
9	0057245cb000da501373ebc645f...	expo-cuisines.com	None	HackedSiteExfil_WebSkimmer	1622157525
10	005aa9c6339c423536ea031aeb3...	gbmglass.com	None	CustomWebSkimmers	1622157522
11	006cf7ffe08fb2c504a36b3b007f6...	elitereplicawatch.is	None	FBseo_WebSkimmer	1622157521
12	006db18e5824162afb1ac5d286d5...	climatecsa.com	None	Infra_WebSkimmer	1623184626
13	0082a7a62463c9bbacf6a3f1836b...	dressaccent.com	None	HackedSiteExfil_WebSkimmer	1622939084
14	00995b98bacb1525356e2d2fd86f...	eurostyleyourlife.com	None	CustomWebSkimmers	1622157519
15	00bd46de3c612c0984382eb94a6...	d5yuj8f7nzjk5.cloudfront.net	None	CustomWebSkimmers	1622157519
16	00bdb1c342efab8afb52c552236f1...	artonfashion.com	None	HackedSiteExfil_WebSkimmer	1622157520
17	00c31ea1de3603c167d7082d7465...	cakeaccessories.co.uk	onlinestatus.site	CustomWebSkimmers	1622157525
18	00fb0e604b1794bdb57053f2c7c4...	oryxchassis.com	None	CustomWebSkimmers	1622157521
19	010940634be919f4b4d628f6db21...	godspeedproject.com	None	CustomWebSkimmers	1622157526

Figure 9: Local web skimmer database.

CONCLUSION

With a diversity of e-commerce platforms and the existential conundrum of applying security updates, attackers are ultimately competing against one another to get the biggest share of vulnerable sites.

Security researchers have a number of tools they can use to discover new Magecart attacks. We showed how someone with a *VirusTotal* licence can identify new victim sites and skimmer gates relying on both anti-virus detections and custom YARA rules.

In fact, the same method presented in this paper can be applied to other data sources as well, provided they expose the right APIs to query against HTML and scripts. This could be done in real time or asynchronously by downloading resource files and then scanning them. As crawling, scanning and storing web traffic requires significant resources, it can be beneficial to tap into existing platforms with a large user base submitting data or URLs regularly. This ensures the best possible coverage for sites big and small powered by a variety of CMS and server applications.

REFERENCES

[1] Ali, F. US ecommerce grows 44.0% in 2020. Digital Commerce 360. 29 January 2021. https://www.digitalcommerce360.com/article/us-ecommerce-sales/.

[2] Klijnsma, Y. Inside the Magecart Breach of British Airways: How 22 Lines of Code Claimed 380,000 Victims. RiskIQ. 11 Septmeber 2018. https://www.riskiq.com/blog/external-threat-management/magecart-british-airways-breach/.

[3] Klijnsma, Y. Inside and Beyond Ticketmaster: The Many Breaches of Magecart. RiskIQ. 9 July 2018. https://www.riskiq.com/blog/external-threat-management/magecart-ticketmaster-breach/.

[4] BBC News. British Airways: Suspect code that hacked fliers 'found'. 11 September 2018. https://www.bbc.com/news/technology-45481976.

[5] Segura, J. Credit card skimmer piggybacks on Magento 1 hacking spree. Malwarebytes. 2 February 2021. https://blog.malwarebytes.com/cybercrime/2021/02/credit-card-skimmer-piggybacks-on-magento-1-hacking-spree/.

[6] VirusTotal. https://www.virustotal.com/gui/search/engines%253AMagecart%2520or%2520engines%253ASkimmer%2520or%2520engines%253ACardStealer%2520or%2520engines%253AMagentoStealer/files.

[7] VirusTotal. Writing YARA rules for Livehunt. https://support.virustotal.com/hc/en-us/articles/360007088057-Writing-YARA-rules-for-Livehunt.

[8] https://twitter.com/search?q=%23magecart&src=typed_query.

[9] urlscan.io. https://urlscan.io/.

[10] Herman, J. A New Grelos Skimmer Reflects the Depth and Murkiness of the Magecart Ecosystem. RiskIQ. 18 November 2020. https://www.riskiq.com/blog/external-threat-management/magecart-grelos/.

[11] Segura, J. New evasion techniques found in web skimmers. Malwarebytes. 30 December 2019. https://blog.malwarebytes.com/threat-analysis/2019/12/new-evasion-techniques-found-in-web-skimmers/.

[12] Segura, J. Web skimmer hides within EXIF metadata, exfiltrates credit cards via image files. Malwarebytes. 25 June 2020. https://blog.malwarebytes.com/threat-analysis/2020/06/web-skimmer-hides-within-exif-metadata-exfiltrates-credit-cards-via-image-files/.

[13] https://community.riskiq.com/.

[14] https://github.com/malwareinfosec/WebSkimmers/blob/main/useful_scripts/make_skimmerinfra_rule.py.

[15] Segura, J. Web skimmer phishes credit card data via rogue payment service platform. Malwarebytes. 21 November 2019. https://blog.malwarebytes.com/web-threats/2019/11/web-skimmer-phishes-credit-card-data-via-rogue-payment-service-platform/.

[16] https://github.com/malwareinfosec/webskimmers.