CVE-2020-1349 Write-Up

Created July 28, 2020 11:13 PM

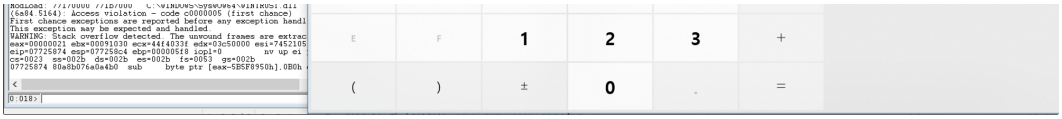
Tags Windows Microsoft Office Outlook Exploit

Author Hangjun Go

CVE-2020-1349

This vulnerability occurs in Outlook 2019 (16.0.12624.20424) installed on Korean Windows 10 1909 x64, codepage 949.

TLDR;



I have not been able to bypass aslr using this vulnerability, but I have written an outlook 2019 32bit exploit that pops up a calculator using the address of a system library that is fixed at boot time. and this vulnerability is a zero click vulnerability. If the account of a mail service other than the “outlook.com” account is linked with Outlook 2019 Using IMAP, it is triggered by receiving mail only.

First Bug

```
v26 = IsDBCSLeadByte(*v13); v12 = 0; if ( v26 ) {
LODWORD(v5) = v5 - 1; *v11++ = *v13++; --v8; } LODWORD(v5)
= v5 - 1; *v11++ = *v13++; --v8; goto LABEL_51;
```

[OUTLMIME!CloseAllSockets+0x50804 Pseudo code]

The crash occurs, and the part where the vulnerability exists is as above. When the string of the parsed To header or From header is Mutibyte, the variable counting the remaining strings to be parsed is decreased by 2, and the integer underflow occurs by decreasing the variable by 2 when the variable is 1.

Second Bug

```
if ( v23 != '=' ) { if ( v23 == 34 ) v15 = 6; goto
LABEL_48; } v15 = 1; } if ( v15 ) goto LABEL_40;
```

[OUTLMIME!CloseAllSockets+0x50804 Pseudo code]

```
LABEL_40: v24 = v13[v14]; if ( v24 ) { if ( v5 == v14 )
goto LABEL_52; if ( v8 != v14 ) { v25 =
```

```
IsDBCSLeadByte(v24); v12 = 0; if ( v25 ) ++v14; ++v14; } }
goto LABEL_51; }
```

[OUTLMIME!CloseAllSockets+0x50804 Pseudo code]

```
LABEL_51: if ( v5 <= v14 ) goto EXIT_0; }
```

[OUTLMIME!CloseAllSockets+0x50804 Pseudo code]

```
if ( v15 ) { if ( v15 == 1 ) { v15 = 2; v21 = v13[v14];
v22 = v14 + 1; if ( v21 != '?' ) v22 = v14; if ( v21 !=
'?' ) v15 = v12; v14 = v22; }
```

[OUTLMIME!CloseAllSockets+0x50804 Pseudo code]

```
else if ( v15 > 1 ) { if ( v15 <= 3 ) { v18 = v13[v14];
v19 = v14 + 1; if ( v18 != '?' ) v19 = v14; v14 = v19; v20
= v15 + 1; if ( v18 != '?' ) v20 = v15; v15 = v20; }
```

[OUTLMIME!CloseAllSockets+0x50804 Pseudo code]

```
if ( v15 == 4 ) { if ( v13[v14] == '?' ) v15 = 5;
LABEL_40: v24 = v13[v14]; if ( v24 ) { if ( v5 == v14 )
goto LABEL_52; if ( v8 != v14 ) { v25 =
IsDBCSLeadByte(v24); v12 = 0; if ( v25 ) ++v14; ++v14; } }
goto LABEL_51; }
```

[OUTLMIME!CloseAllSockets+0x50804 Pseudo code]

```
if ( v15 == 5 ) { v16 = v14; v17 = v14 + 1; v30 =
v13[v16]; if ( v30 == '=' ) { *v11++ = 34; --v8; }
memcpy(v11, v13, v17); v13 += v17; v11 += v17; LODWORD(v5)
= v5 - v17; v8 -= v17; if ( v30 == '=' ) { *v11++ = 34;
--v8; } v12 = 0; v14 = 0; v15 = 0; goto LABEL_51; }
```

[OUTLMIME!CloseAllSockets+0x50804 Pseudo code]

```
else { v23 = v13[v14]; if ( v23 != '=' ) { if ( v23 == '"'
) v15 = 6; goto LABEL_48; } v15 = 1; } if ( v15 ) goto
LABEL_40;
```

[OUTLMIME!CloseAllSockets+0x50804 Pseudo code]

```
LABEL_48: v26 = IsDBCSLeadByte(*v13); v12 = 0; if ( v26 )
{ LODWORD(v5) = v5 - 1; *v11++ = *v13++; --v8; }
LODWORD(v5) = v5 - 1; *v11++ = *v13++; --v8; goto
LABEL_51;
```

[OUTLMIME!CloseAllSockets+0x50804 Pseudo code]

However, when triggering a heap overflow using the above bug, I couldn't satisfy the condition of getting out of the loop again, so I couldn't perform the partial overwrite as long as I wanted. So I had to find an input that satisfies a specific condition that would go out of the loop. For the same reason as above, I strictly audited the surrounding code part, and instead of finding the input value that can escape the loop, the bug of falling into an infinite loop without increasing the index or writing when processing a special string.

EIP Control

```
WARNING: Continuing a non-continuable exception
(69f8.6b80): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=0c0c0c0c ebx=00000001 ecx=0495fc98 edx=737b0000 esi=74836590 edi=00000000
eip=0c0c0c0c esp=11c9fbd8 ebp=11c9fbe4 iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010202
0c0c0c0c 33582b          xor     ebx,dword ptr [eax+2Bh] ds:002b:0c0c0c37=18c5e9fa
```

After performing heap spray and heap feng shui, eip hijacking was succeeded by overwriting the vftable of the running object in another thread that was not caught in an infinite loop using the above bug.