Malware Analysis Report
(Macro_Agent_Dropper)

S.NO	Analysis File	File Type	Analysis start Date	Analysis End date
1	Dort_act_15880_25.doc	Microsoft Word	02-24-2020	02-25-2020

Analysis Performed by,
Sreeharsha Bandi

Executive summary

The document consists of VB script embedded inside it, which is called as a macro. When a user opens the word document and click on the enable content option, the macro embedded inside it will start executing automatically and at the same time the macro starts writing a new Java_script_encrypted (.jse) file. The generated .jse file is completely obfuscated. This obfuscated file consists of a C&C server address, from where the further droppers are being downloaded.
The technique used in this process is **ostap,** which is a JavaScript downloader and is used to bypass security controls.

OSTAP:

Ostap is a commodity JScript downloader first seen in campaigns in 2016. It has been observed being delivered in ACE archives and VBA macro-enabled Microsoft Office documents. Recent versions of Ostap query WMI to check for a blacklist of running processes:

- AgentSimulator.exe
- anti-virus.EXE
- BehaviorDumper
- BennyDB.exe
- ctfmon.exe
- fakepos_bin
- FrzState2k
- gemu-ga.exe (Possible misspelling of Qemu hypervisor's guest agent, qemu-ga.exe)
- ImmunityDebugger.exe
- KMS Server Service.exe
- ProcessHacker
- procexp
- Proxifier.exe
- python
- tcpdump
- VBoxService
- VBoxTray.exe
- VmRemoteGuest
- vmtoolsd
- VMware2B.exe
- VzService.exe
- winace
- Wireshark

OLE FORMAT:

Object Linking & Embedding (OLE) is a proprietary technology developed by Microsoft that allows embedding and linking to documents and other objects. These objects are used to write a script application to the disk that facilitates the download and execution of a malware payload. Malware authors are now using OLE embedding to deliver malicious files.

IOC

Files Opened

- C:\users\binary\appdata\roaming\microsoft\Dsaow.GaerIok
- C:\Windows\System32\WScript.exe
- C:\Users\binary\AppData\Local\Temp\VBE
- C:\Program Files\Common Files\Microsoft Shared\VBA\VBA7.1\VBE7.DLL
- C:\WINDOWS\splwow64.exe

Created files

- C:\users\binary\appdata\roaming\microsoft\Dsaow.GaerIok.jse

Modified registers

- HKEY_CURRENT_USER\Software\Microsoft\VBA\7.1\Common\CodeForeColors
- HKEY_CURRENT_USER\Software\Microsoft\VBA\7.1\Common
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows Script Host\Settings\Enabled
- HKEY_CURRENT_USER\Software\Microsoft\VBA\7.1\Common\BackGroundCompile
- HKEY_CURRENT_USER\Software\Microsoft\VBA\7.1\Common\OBGroupMembers
- HKEY_CLASSES_ROOT\.jse
- HKEY_CURRENT_USER\Software\Microsoft\Windows Script Host\Settings\Enabled
-

Mutexes

- Dsaoyyyw.GayyyerIok.

Created processes

- c:\program files\microsoft office\root\office16\winword.exe
- c:\windows\system32\wscript.exe
- c:\windows\splwow64.exe

Execution process

Winword.exe ---- > splwow64.exe ------ > wscript.exe (for maintaining persistence and executing the obfuscated code)

Note:

Normally wscript.exe closes automatically after usage, but during the analysis it was observed that wscript.exe keeps on running in the background and interacting with the obfuscated code, which was written by the macro and this is also looking for the startup application, WMI and Mstsc to create persistence. If the persistence is successful then the attacker can convert the system into a bot and do further attacks.

Network communication

- http://185.180.199.77/3mBhb0/6VIJ7e.php?d=

Complete analysis

STEP 1:

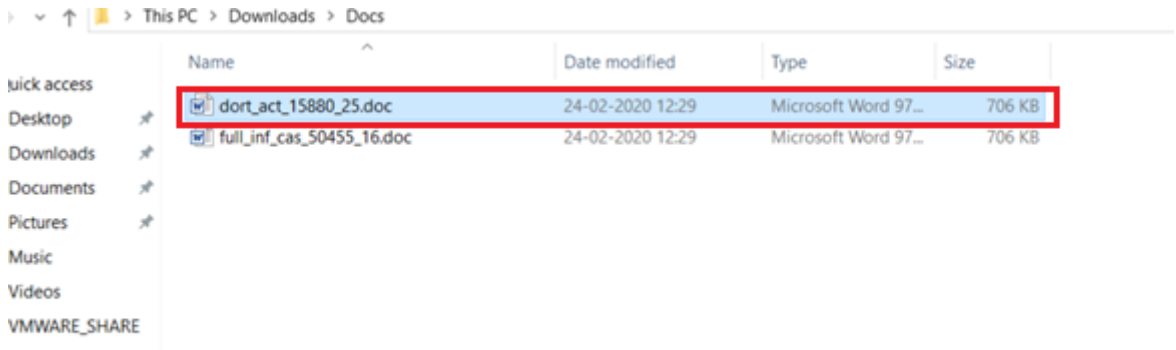


Fig.1: Analysis sample.

STEP 2: Static analysis

File name:	*Dort_act_15880_25.doc*
File size:	706 KB
File type:	Microsoft Word
MD5:	6cb29be017c9a0d5fb636dbda5a772da
SHA1:	b1736e88301757ff9da805b9c4f9259311449125
SHA256:	570b35cc8e93412628804445939bc6ea480dc42c97bd409ee7517bf6124cf7e9
SSDeep:	6144:dhcAB66/16FlBvZjhrQzdcuOFX1y7R5U0jfkTmHRRfspL/7OyBnb1MIibfecUg:L91mEzdcuA1y7k0ZfA/7vJ6bfR

- The analyzed sample is a Microsoft Word macro-enabled document. The VBA macro can be extracted using the tool olevba from python package oletools. The extracted source code of macro is shown in the screenshots below.

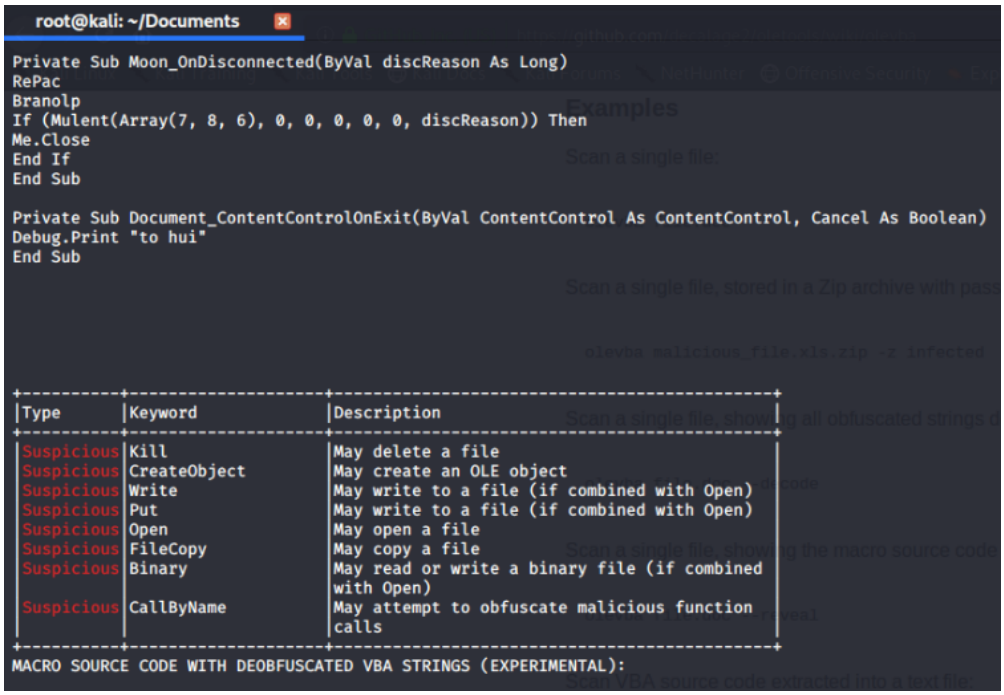


Fig. 2: OLE stream extracted data

Fig.3: VB macro code.

STEP 3: Dynamic analysis

- Analysis machine: Windows 7 x64, Windows 10 x64 1903 – VMs

Note: Analysis is performed with and without (internal network) network connectivity.

Tools Used:

- Procmon
- Process explorer
- InetSim
- Oletools
- IDA
- Wireshark

- Regmon
- Ostap jse DE obfuscation.

Virtualization Software:

- VMware workstation Pro
- VirtualBox

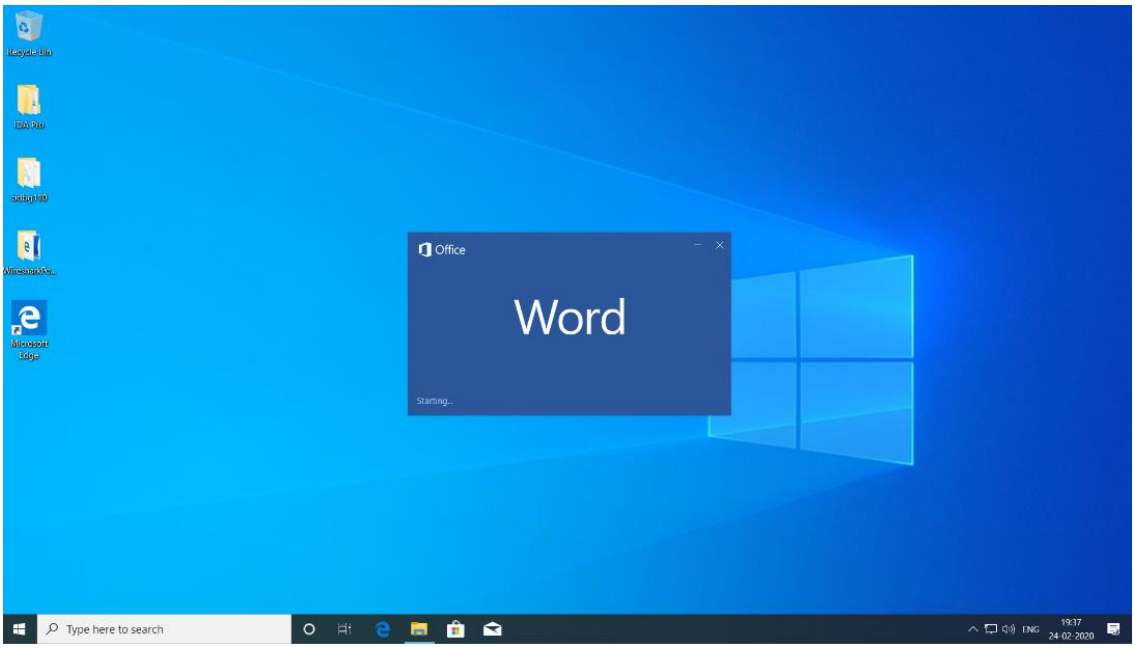


Fig. 4: Executing the sample

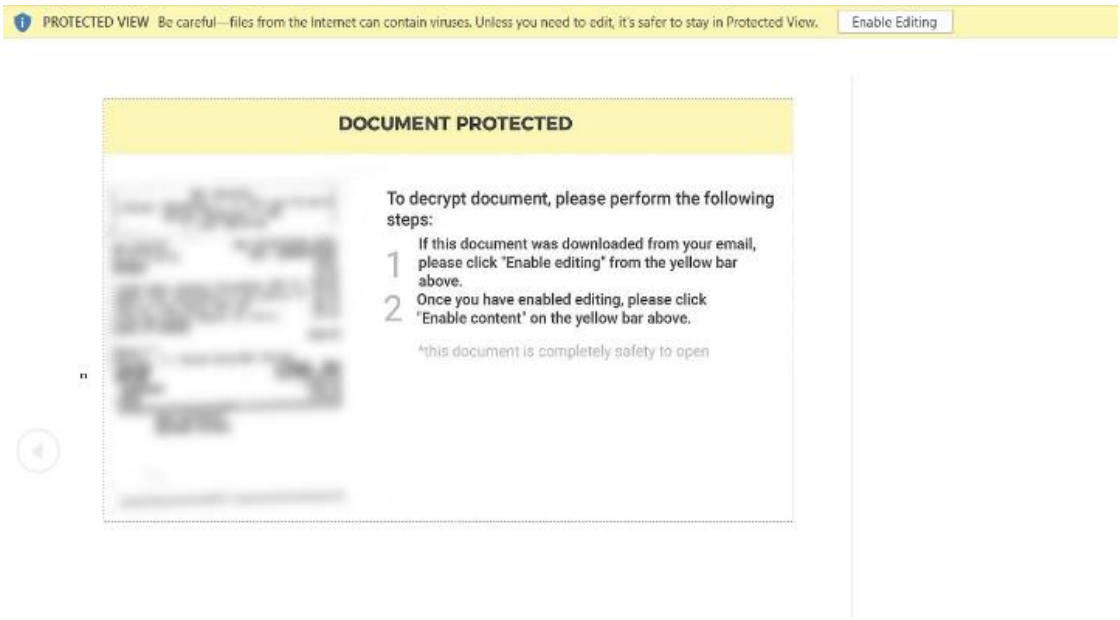


Fig.5: Enable editing notification.

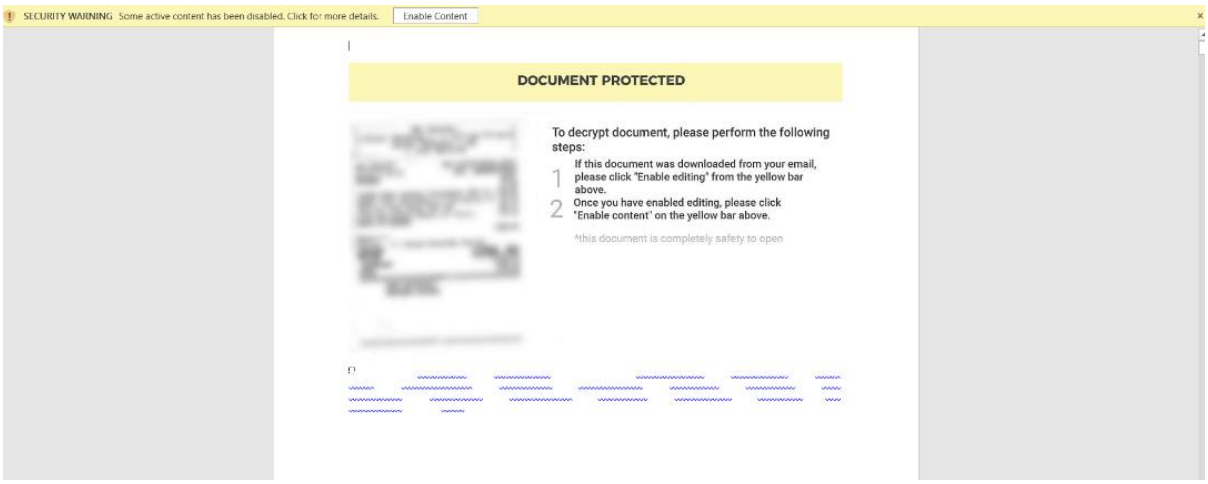


Fig.6: Enable content notification.

Note: In order to run the macro inside the document, it tricks the user to click on enable editing and enable content options. This type of techniques is called as Trickbot, which acts as a benign sample.

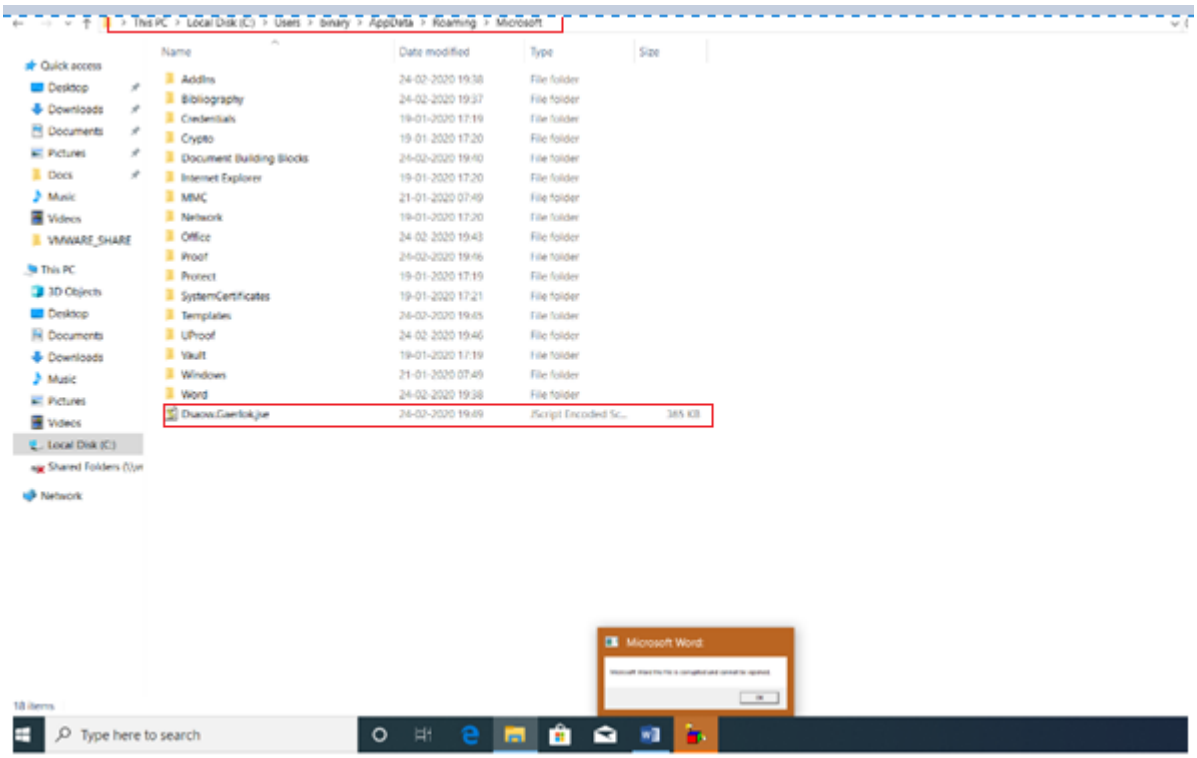


Fig.7: encoded JavaScript created after executing the document.
(PATH: users\xxxx\Appdata\Roaming\Microsoft\Dsaow.Gaerlok.jse)

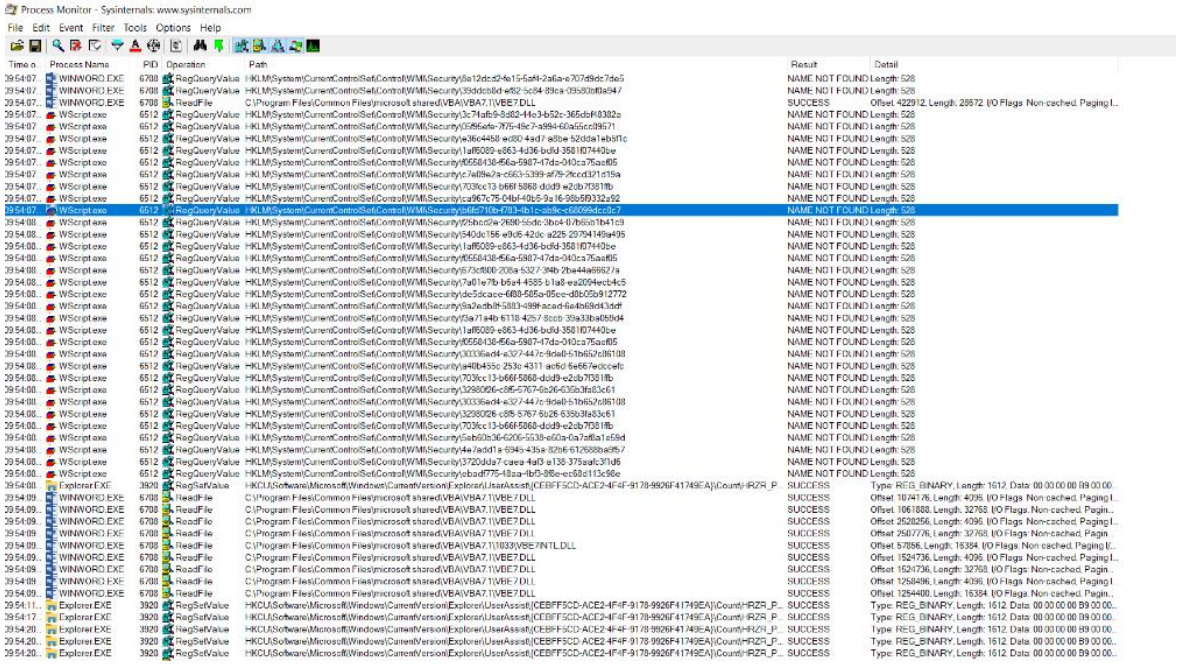


Fig.8: Trying to create New Registry values and also connecting to WMI for persistence.

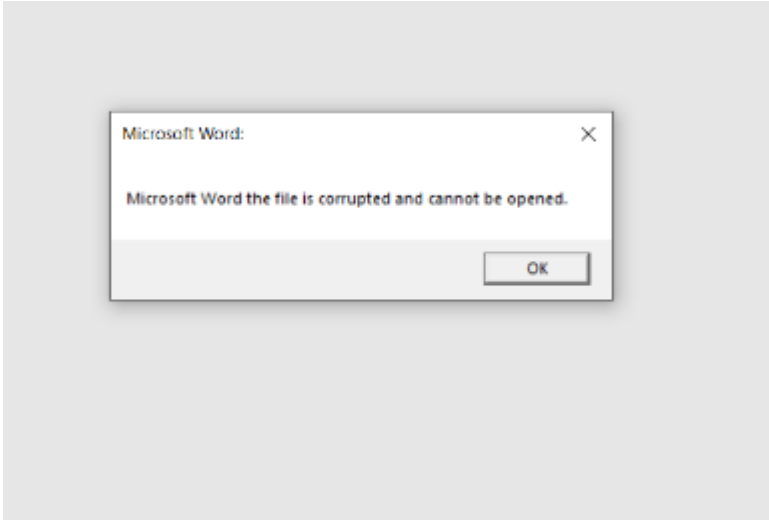
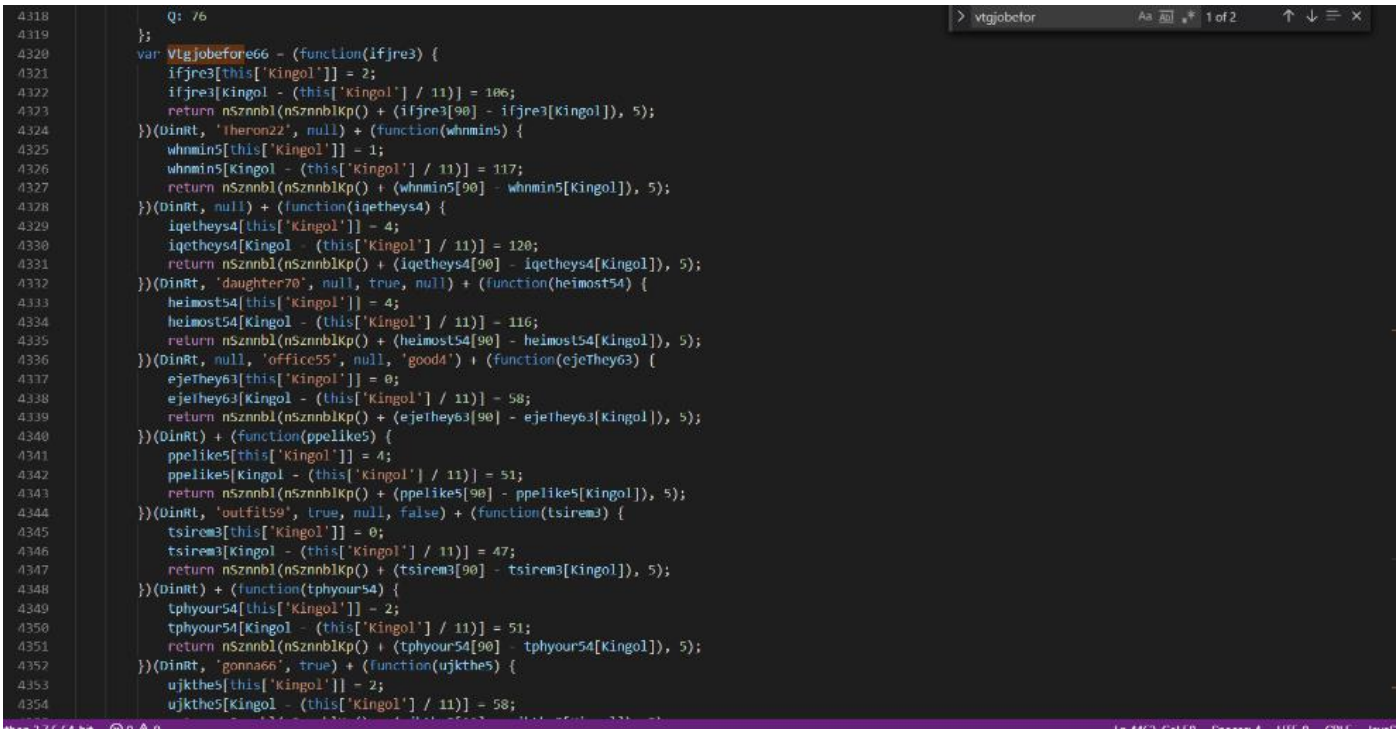


Fig.9: After performing the required action the macro is automatically closing.

- Even after closing the word document, the wscript.exe continues to run in the background and interacts with the .jse file, which has the URL of the C&C server. Then from the C&C server it downloads the further droppers to do more harm to the user.

Decoding the JavaScript code:



```
        Q: 76
    };
    var Vtgjobefore66 = (function(ifjre3) {
        ifjre3[this['Kingol']] = 2;
        ifjre3[Kingol - (this['Kingol'] / 11)] = 106;
        return nSznnbl(nSznnblKp() + (ifjre3[90] - ifjre3[Kingol]), 5);
    })(DinRt, 'Theron22', null) + (function(whnmin5) {
        whnmin5[this['Kingol']] = 1;
        whnmin5[Kingol - (this['Kingol'] / 11)] = 117;
        return nSznnbl(nSznnblKp() + (whnmin5[90] - whnmin5[Kingol]), 5);
    })(DinRt, null) + (function(iqetheys4) {
        iqetheys4[this['Kingol']] = 4;
        iqetheys4[Kingol - (this['Kingol'] / 11)] = 120;
        return nSznnbl(nSznnblKp() + (iqetheys4[90] - iqetheys4[Kingol]), 5);
    })(DinRt, 'daughter70', null, true, null) + (function(heimost54) {
        heimost54[this['Kingol']] = 4;
        heimost54[Kingol - (this['Kingol'] / 11)] = 116;
        return nSznnbl(nSznnblKp() + (heimost54[90] - heimost54[Kingol]), 5);
    })(DinRt, null, 'office55', null, 'good4') + (function(ejeThey63) {
        ejeThey63[this['Kingol']] = 0;
        ejeThey63[Kingol - (this['Kingol'] / 11)] = 58;
        return nSznnbl(nSznnblKp() + (ejeThey63[90] - ejeThey63[Kingol]), 5);
    })(DinRt) + (function(ppelike5) {
        ppelike5[this['Kingol']] = 4;
        ppelike5[Kingol - (this['Kingol'] / 11)] = 51;
        return nSznnbl(nSznnblKp() + (ppelike5[90] - ppelike5[Kingol]), 5);
    })(DinRt, 'outfit59', true, null, false) + (function(tsirem3) {
        tsirem3[this['Kingol']] = 0;
        tsirem3[Kingol - (this['Kingol'] / 11)] = 47;
        return nSznnbl(nSznnblKp() + (tsirem3[90] - tsirem3[Kingol]), 5);
    })(DinRt) + (function(tphyour54) {
        tphyour54[this['Kingol']] = 2;
        tphyour54[Kingol - (this['Kingol'] / 11)] = 51;
        return nSznnbl(nSznnblKp() + (tphyour54[90] - tphyour54[Kingol]), 5);
    })(DinRt, 'gonna66', true) + (function(ujkthe5) {
        ujkthe5[this['Kingol']] = 2;
        ujkthe5[Kingol - (this['Kingol'] / 11)] = 58;
```

Fig.10: Encoded JavaScript code after beautifying the code using JS beautifier.

Code logic:

```
var Vtgjobefore66 = (function(ifjre3) {
        ifjre3[this['Kingol']] = 2;
        ifjre3[Kingol - (this['Kingol'] / 11)] = 106;
        return nSznnbl(nSznnblKp() + (ifjre3[90] - ifjre3[Kingol]), 5);
    })(DinRt, 'Theron22', null) + (function(whnmin5) {
        whnmin5[this['Kingol']] = 1;
        whnmin5[Kingol - (this['Kingol'] / 11)] = 117;
        return nSznnbl(nSznnblKp() + (whnmin5[90] - whnmin5[Kingol]), 5);
    })(DinRt, null) + (function(iqetheys4) {
iqetheys4[this['Kingol']] = 4;
```

- From the above code the function Vtgjobfore66 is similar to FromCharCode:
  ifjre3[this['Kingol']] = 2; ==== consider as “a”
  ifjre3[Kingol - (this['Kingol'] / 11)] = 106; ==== consider as “b”

  c = b-a (i.e., 106-2 = 104)
  char letter = convert_to_char_code(c) (i.e charcode(104) == ‘h’)

Similarly, for all the subfunctions: we get charcode(104, 116, 116, 112, 58, 47, 47, 49, 56, 53, 46, 49, 56, 50, 46, 49, 57, 57,46,55,55,47,51,109,66,104,98,48,47,54,86,73,74,55,101)
Which gives result as ::::::::: http://185.180.199.77/3mBhb0/6VIJ7e.php?d=

Charcode Script:

```
<script>
function myFunction() {
  var res = String.fromCharCode(104, 116, 116, 112, 58, 47, 47, 49, 56, 53, 46, 49, 56, 50, 46, 49, 57,
    57,46,55,55,47,51,109,66,104,98,48,47,54,86,73,74,55,101);
  document.getElementById("demo").innerHTML = res;
}
    </script>
```

Automation using regex:

```
import re
samples = []

with open('code.js') as myfile:
    for line in myfile.readlines():
        if re.search(r'/(?<=\[kingol-\(this[kingol]\/11)]=)[^;]+/g', line):
            samples.append(line)

# print('SAMPLES: ', samples)

with open("file2.txt", "w") as myfile2:
    for s in samples:
        myfile2.write(s)

#/(?<=\[kingol-\(this['kingol']\/11)]=)[^;]+/g
    #(?<=\[this['Kingol']=)[^;]+/g

import re
import sys

def deobfuscate(s):
      pattern =
r"""\s\+\sfunction\s\(\)\s\{\s+var\s.*?\s=\s.*?;\s+.*?\[.*?\]\s=\s(?P<first>\d+);\s+.*?\[\d+\]\s=\s(?P<se
cond>\d+);\s+return.*?\(.*?, 'a'\);\s+\}\(.*?\)"""
      while re.findall(pattern,s):
            r = re.findall(pattern,s)[0]
            s = (re.sub(pattern,chr(int(r[0]) + int(r[1])),s,1))

      pattern =
r"""function\s\(\)\s\{\s+var\s.*?\s=\s.*?;\s+.*?\[.*?\]\s=\s(?P<first>\d+);\s+.*?\[\d+\]\s=\s(?P<second
>\d+);\s+return.*?\(.*?, 'a'\);\s+\}\(.*?\)"""
      while re.findall(pattern,s):
            r = re.findall(pattern,s)[0]
            s = (re.sub(pattern,chr(int(r[0]) + int(r[1])),s,1))       print(s)
```

Note:

For code de obfuscation, ostap jse python script can also be used if the array indexes are clearly mentioned.

Further Process:

- Checks whether the running script is in %TEMP% or Roaming folder by searching for the substring "\temp" in WScript[ScriptFullName].
- If the running script is not in %TEMP%, the sample produces an error message popup, copies the contents of the document to a variable and appends "var seed<random_integer>=<random_integer>;" to the variable.
- Uses WMI tasks to fingerprint Win32_Operating System, Win32_ComputerSystem, and Win32_Process Operating System Classes data.
- POSTs fingerprint to C2
- These WMI task fingerprinting techniques have been associated with OSTAP droppers in the past, which indicates this is an artifact from older samples.
- Acquires a positive random integer smaller than 2^mod(c-7), which it uses as a .txt filename and a "&z=" GET parameter.
- Saves a copy of the white-font hidden JScript from the existing variable (with the appended seed) to the random integer named text file (which we will now call persistence.txt).
- Creates an .LNK shortcut file with filename maxp.lnk to the Windows Startup folder.
- The .LNK file has a target path of: WScript, and arguments: /B /e:Jscript <path to persistence.txt>
- This technique is used by attackers to persist upon shutdown and restart.

END OF REPORT