'I CAN'T GO BACK TO YESTERDAY, BECAUSE I WAS A DIFFERENT PERSON THEN'[1]

Chun Feng
Microsoft, Level 5, 4 Freshwater Place,
Southbank, Victoria, 3006, Australia

Email chfeng@microsoft.com

ABSTRACT

System Restore hardware and software have been widely implemented, and are commonly used by computer users to revert back to a pre-preserved 'good' state after being affected by malware or other threats to system integrity. As these restore facilities have become commonplace, so too has the malware that attempts to penetrate them. This type of malware reaches into the depths of the affected machine and targets the file system driver.

In late 2007, a mysterious new breed of malware appeared in China and has been evolving quickly since. This malware, named Win32/Dogrobot, is designed deliberately to penetrate a 'hard disk recovery card' – hardware widely used by Internet cafés in China. Surprisingly, Dogrobot has caused more than eight billion RMB (around 1.2 billion USD) in losses to Internet cafés in China. (This cost far exceeds that caused by the notorious Win32/Viking virus.)

This paper tracks the five generations of Dogrobot and presents the novel rootkit technique used by Dogrobot to penetrate System Restore on *Windows* systems, covering penetration from the *Windows* volume management layer used by early variants, to the *Windows* IDE/ATAPI Port Driver layer used by the latest variants. This paper also closely examines Dogrobot's propagation methods, including the use of zero-day exploits and ARP spoofing.

What is the significance of Dogrobot's selection of Internet cafés as its chosen targets? And what is the final goal of this malware? This paper answers these questions and elaborates on the clandestine relationship between Dogrobot and the black market for online games passwords.

INTRODUCTION

There are a number of reasons why computer users and administrators might need to revert their system back to a previously known good state, e.g. when recovering from a system failure, cleaning a malware infection or during regular maintenance. These scenarios tend to occur more often in public and academic environments, like Internet cafés or computer laboratories. To make tracking and reverting of the changes easier, vendors devised System Restore facilities (hereafter referred to as System Restore) to automate these preservation and reversion processes. System Restore has been widely implemented in both hardware (e.g. HD Recovery Card [1]) and software solutions (e.g. Drive Vaccine [2]). It has even been built into operating systems (e.g. *Windows XP* and *Windows Vista*). System Restore monitors and tracks system changes in the background (invisible to users), and allows users to revert back to a saved restoration point, either on demand, or automatically according to defined rules, e.g. after each *Windows* reboot, or once every month.

System Restore has more than a 20-year history and has been considered a good solution for protecting and maintaining system integrity, especially for Internet cafés. However, this was changed by a storm caused by some malware named Dogrobot, which appeared in China in late 2007 and has been evolving since. This new breed of malware has been designed to compromise the protection provided by a hard disk recovery card, the solution which is widely used by Internet cafés in China. The Dogrobot family is estimated to have caused more than eight billion RMB (around 1.2 billion USD) in losses to Internet cafés in China – a cost that far exceeds that caused by other, more notorious malware: Win32/Viking [3].

In this paper, we examine current designs and implementations of System Restore, and analyse techniques used by five generations of Dogrobot to penetrate these systems. We also closely examine Dogrobot's propagation method. Finally, we reveal Dogrobot's purpose and elaborate on the relationship between Dogrobot and the black market of stolen online games passwords [4].

I CAN GO BACK TO YESTERDAY

Designs and implementations of System Restore may be approached in different ways: The *Windows* operating system (*Windows XP* or *Windows Vista*) implements System Restore at a file system layer; another solution is to use a lower layer (disk layer). The mechanism of disk layer System Restore is illustrated in Figure 1[2] [5].

System Restore reserves a certain amount of space, known as 'scratch space', from the free disk space. The disk space is virtually split into three parts:

- Permanent storage, which stores the permanent data – usually System Restore protects this space as 'read-only' for users; it can only be written to by System Restore, e.g. when the user chooses to commit changes.
- Scratch space, which stores the volatile data – any system change takes place in this space before being committed.
- Free space, which is not used and can be allocated for future use.

System Restore mirrors the original FAT (File Allocation Table) or MFT (Master File Table) from permanent storage into the scratch space as FAT´ or MFT´. Then it adapts the following rules for disk-level read/write operations:

- Any disk read/write operation from/to FAT will be re-routed to FAT´.
- An attempt to write to file X makes System Restore allocate some space from the scratch area, write data to a file X´ and then update FAT´ to reflect the change. The original file pointer to X now points to X´.
- An attempt to read from an already modified file e.g. file X, causes the system to read from the scratch space (X´) instead. Reading unmodified files, e.g. file Y, proceeds as usual from the original space, i.e. permanent storage.

[1] 'I can't go back to yesterday – because I was a different person then' Lewis Carroll.

[2] This is for illustration purposes only. Some technical details are omitted here for the sake of simplicity; it could be more complicated in reality.

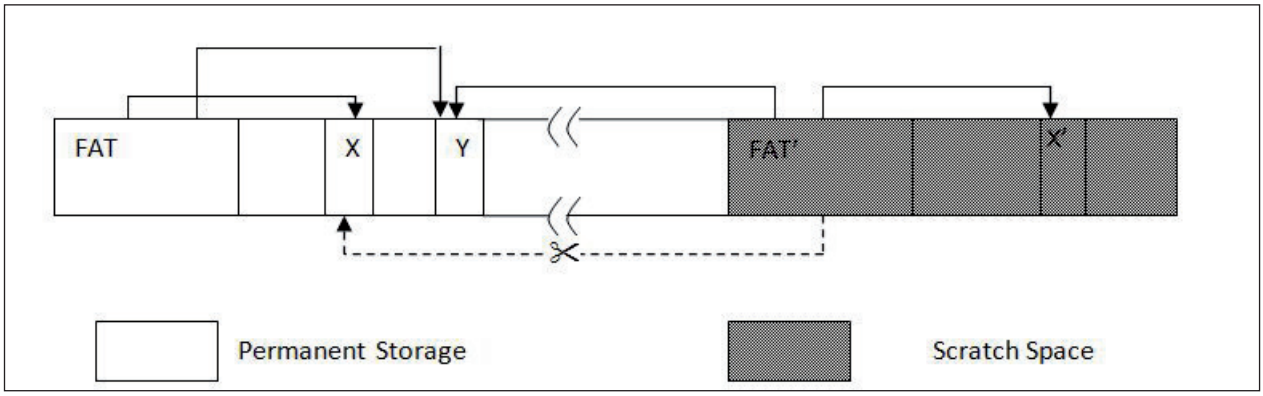

Figure 1: *The re-routing mechanism of System Restore [5].*

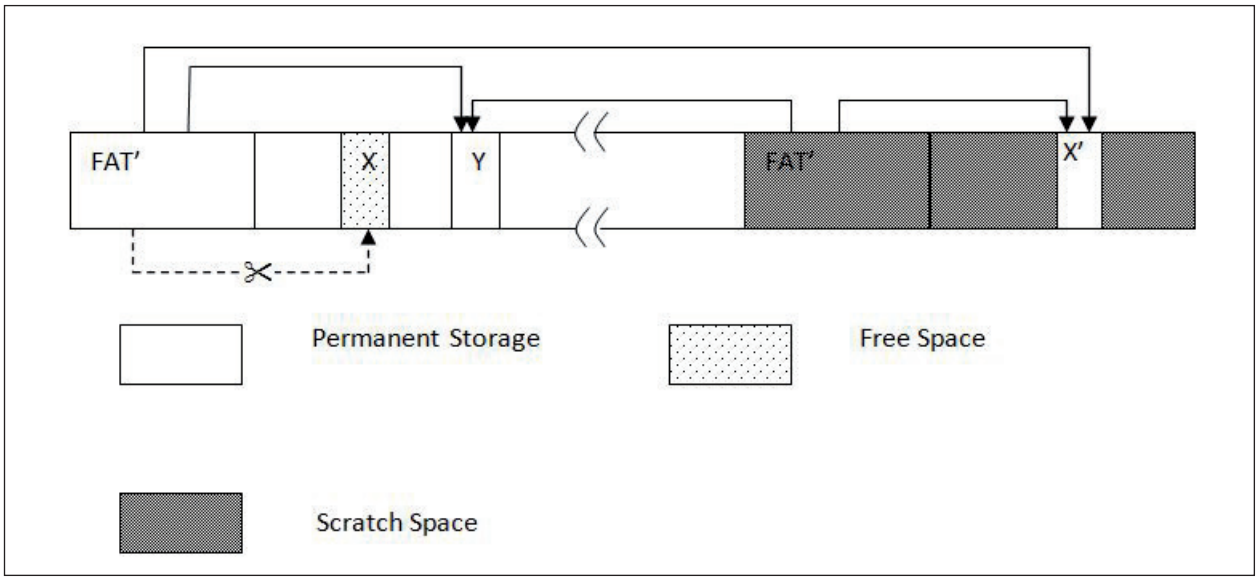

Figure 2: The re-routing mechanism of System Restore facilities (after commit).

- When reverting, System Restore simply overwrites the FAT´ from the area with the original content of FAT (all changes saved in the scratch space are discarded).
- When committing, System Restore just copies FAT´ back to FAT (all changes stored in the scratch space are copied to permanent storage, see Figure 2).

In *Windows* systems, System Restore usually implements a filter driver, which is attached to a disk device: \Device\Harddisk0\DR[N] (created by the device driver: disk.sys). Since the *Windows* I/O manager always dispatches the IRP (I/O request packet) to the topmost device in the device stack before it is passed down to the next lower device in the stack, all IRPs will be received by the disk filter driver first. This means that System Restore can intercept and modify a disk read/write IRP before it reaches the disk driver (disk.sys).

I CAN'T GO BACK TO YESTERDAY

As System Restore solutions become popular, so too does malware that attempts to penetrate such protection in order to survive the process of reverting a compromised system to its previous clean state. Dogrobot uses disk-level I/O file manipulation to penetrate System Restore, although the methods implemented by its five generations are variable.

First generation

Samples of the first generation of Dogrobot were observed in the wild in September 2007. They were packed with their own packer, both PE image sections were named 'SEC', and they used the *SONY AIBO* picture as an icon. The latter explains the origin of the name assigned to this malware family (see Figure 3).

Figure 3: Icon used by the first generation of Dogrobot.

This malware's functionality can be summarized in the following five major steps:

1. Dogrobot drops and loads the device driver pcihdd.sys. The driver saves the value of DEVICE_OBJECT-> AttachedDevice of the device object \Device\Harddisk0\DR0 so it can be restored in the future, and then it clears it (sets it to NULL) (see Figure 4).
2. Dogrobot, in user mode, locates the start offset of the file userinit.exe on the disk by sending I/O control code FSCTL_GET_RETRIEVAL_POINTERS to the file system. It reads 512 bytes (one sector) from this disk offset and compares it with the result returned by a

```
push   offset SourceString ; "\\Device\\Harddisk0\\DR0"
lea    eax, [ebp+DestinationString]
push   eax        ; DestinationString
call   RtlInitAnsiString
push   1 ; AllocateDestinationString
lea    eax, [ebp+DestinationString]
push   eax        ; SourceString
lea    eax, [ebp+ObjectName]
push   eax        ; DestinationString
call   RtlAnsiStringToUnicodeString
xor    eax, eax
mov    [ebp+FileObject], eax
mov    [ebp+DeviceObject], eax
lea    eax, [ebp+DeviceObject]
push   eax        ; DeviceObject
lea    eax, [ebp+FileObject]
push   eax        ; FileObject
push   80h        ; DesiredAccess
lea    eax, [ebp+ObjectName]
push   eax        ; ObjectName
call   IoGetDeviceObjectPointer
mov    eax, [ebp+FileObject]
jmp    short $+2
mov    eax, [eax+FILE_OBJECT.DeviceObject]
mov    g_devDR0, eax
cmp    [eax+DEVICE_OBJECT.AttachedDevice], 0
jz     short loc_40059A
jmp    short $+2
mov    ecx, [eax+DEVICE_OBJECT.AttachedDevice]
xchg   ecx, g_savedAttachedDevice
mov    [eax+DEVICE_OBJECT.AttachedDevice], ecx
```

Figure 4: Dogrobot's routine to detach the attached device for \Device\Harddisk0\DR0.

ReadFile() call from userinit.exe, to make sure the offset is correct.

3. Again in user mode, the trojan passes its own code section to the driver via the DeviceIoControl API using the I/O control code 0xf0003c04. The driver then obtains a key from the received buffer and decrypts the data contained in its resource. Next, it passes the decrypted data (4KB) back to the user-mode application.
4. Still in user mode, the trojan writes the decrypted payload to the first cluster of the file userinit.exe by direct access to \\.\PhysicalDrive0, which is the symbolic link to \Device\Harddisk0\DR0.
5. Dogrobot restores the saved value of DEVICE_OBJECT->AttachedDevice of \Device\Harddisk0\DR0.

In step 1), the device attached to \Device\Harddisk0\DR0 is cleared. This makes \Device\Harddisk0\DR0 the topmost device in the stack, which means the disk filter driver used by System Restore is not valid in this device stack. Thus System Restore can't intercept the IRP sent directly to \Device\Harddisk0\DR0 any more, i.e. if the user-mode application sends IRP directly to \Device\Harddisk0\DR0, System Restore is bypassed. When, in step 3), the trojan passes the loaded code

```
PSCSI_REQUEST_BLOCK srb
srb->SrbFlags= SRB_FLAGS_DATA_OUT;
srb->SrbFlags|=SRB_FLAGS_ADAPTER_CACHE_ENABLE ;
srb->Length=sizeof(SCSI_REQUEST_BLOCK);
srb->DataBuffer=buffer;
srb->DataTransferLength= bufferLength; // in bytes
// srb->Cdb initialization for disk offset etc.
```

Figure 5: Using SCSI_REQUEST_BLOCK to perform disk write.

section to the driver, this operation constitutes an anti-debugging trick. Any soft breakpoint set within this code area by an analyst or a breakpoint set by a debugger for single-step debugging, may cause an incorrect byte (0xCC) to be passed to the driver, which would result in incorrect decryption. In step 4), the trojan writes data to the file userinit.exe at disk level, so the timestamp and the file size of userinit.exe won't change.

Second generation

Second-generation samples of Dogrobot were first seen in the wild in February 2008, five months after the appearance of the first generation. The second generation uses a different technique to penetrate System Restore. Unlike the first generation of Dogrobot it uses a 'backdoor' that exists in System Restore. Most System Restore solutions intercept IRP_MJ_READ and IRP_MJ_WRITE requests only, and don't handle IRP_MJ_INTERNAL_DEVICE_CONTROL requests. On the *Windows* operating system, an IRP_MJ_INTERNAL_DEVICE_CONTROL IRP request will be forwarded by disk.sys to a lower disk driver directly, and the user application can deliberately construct an SCSI_REQUEST_BLOCK structure to perform disk sector read/write operations. The code for disk writing via SCSI_REQUEST_BLOCK is outlined in Figure 5 [6].

Another change in the second generation of Dogrobot is that the malware doesn't parse the file system itself to find the disk offset of a file; it uses a 'virtual disk' technique with the offset being calculated by the operating system. Analysis of the 'virtual disk' code shows that it is clearly based on the source code of FileDisk [7]. Second-generation samples create 26 disk devices (from \Device\zzz\zzz0 through to \Device\zzz\zzz25) and 26 symbolic links (from \Global??\yyy0 to \Global??\yyy25) for each of these disk devices. The disk devices and symbolic links are created for each possible system drive from A to Z (see Figure 6).

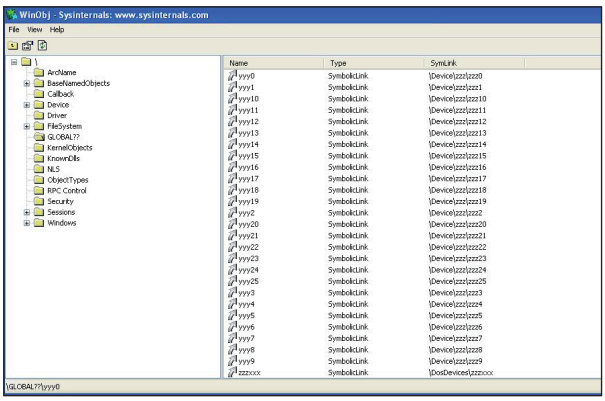

Figure 6: Devices and symbolic links created by second generation of Dogrobot.

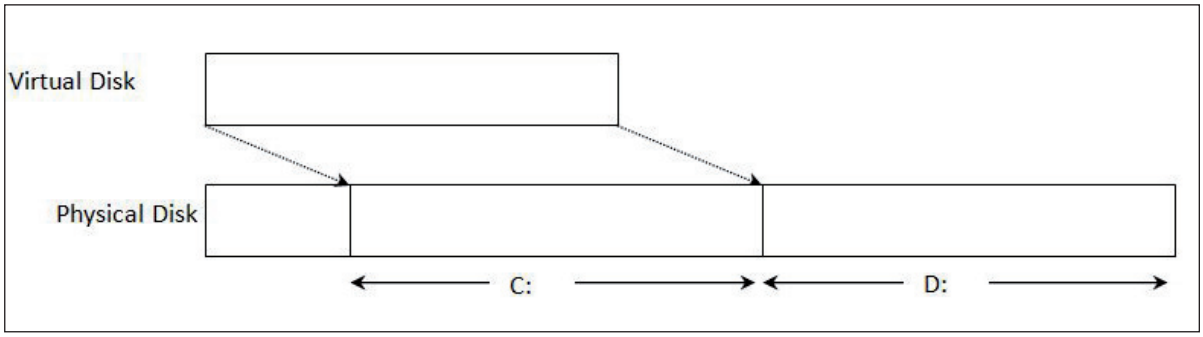

Figure 7: The mapping from virtual disk to physical disk used by the second Dogrobot generation sample.

Assuming the system drive is drive C, Dogrobot uses the disk device \Device\zzz\zzz2 and the symbolic link \Global??\yyy2. The virtual disk read/write operations are mapped to the operations on volume \Device\HarddiskVolume1 (i.e. drive C) via the backdoor mentioned earlier: IRP_MJ_INTERNAL_DEVICE_CONTROL request (see Figure 7). After this virtual disk is mounted as a volume by I/O manager, Dogrobot is able to manipulate files on this virtual disk and, as a result, on the physical drive C.

Compared with the first generation, the virtual disk technique used by the second generation is an 'improvement' in the following respects:

- It is more reliable since it doesn't detach (even temporarily) the DEVICE_OBJECT->AttachedDevice of \Device\Harddisk0\DR0, which may degrade system integrity.
- It makes it much easier for a user-mode application to manipulate files; it doesn't need to parse the disk offset of the file like the first generation (this will be done by the operating system since the virtual disk has been mounted by the file system). The second generation simply penetrates System Restore and copies itself to the virtual disk using the following function call: CopyFile('\\.\yyy2\Documents and Settings\All Users\Start Menu\Programs\Startup\Atisrv.exe', 'dogrobot.exe')[3]. In the first generation, it was harder to break the protection by creating a new file rather than overwriting an existing file, since creating new files modifies the FAT/MFT.
- It provides better compatibility. The first generation doesn't work if the system drive is a compressed drive, while the second generation can handle this case since it manipulates the files through the file system. The first generation contains hard-coded parsing code for FAT, FAT32 and NTFS systems only, while the second generation can handle any file system as long as it is supported by the *Windows* operating system.

In the second generation, the payload is not encrypted in the driver any more; presumably the author attempts to make it easier to generate samples in large volume.

Third generation

Third generation samples of Dogrobot were first noticed in the wild in June 2008. The code of the third generation is based on the improved first-generation code. It is unclear why the Dogrobot author abandoned the virtual disk technique used by the second generation. The malware author appears to have analysed some of the System Restore code as well as several detection/removal programs. This resulted in the introduction of extensive unhooking code to thwart the protection offered by these programs and hinder Dogrobot's detection and removal. The third generation functionality is described in the following five steps [8]:

1. The trojan detaches the DEVICE_OBJECT->AttachedDevice of the NTFS file system device object (\FileSystem\Ntfs), so any filter driver belonging to anti-malware software or System Restore solutions will be disabled.
2. Apart from detaching DEVICE_OBJECT->AttachedDevice of \Device\HardDisk0\DR0 it also removes the hook from the IRP_MJ_INTERNAL_DEVICE_CONTROL dispatch function in the disk.sys driver (driver object \Driver\Disk).

   The third generation accomplishes the unhooking using the following method: it enumerates the structure _LDR_DATA_TABLE_ENTRY pointed to by DRIVR_OBJECT->DriverSection until it finds the module classpnp.sys, which contains the original IRP__MJ_* function's address in disk.sys. Then it checks the RVA of the entry point in this module and, if the value is 0xAE8F, it sets the dispatch function for IRP_MJ_INTERNAL_DEVICE_CONTROL to the value:

   <Imagebase of classpnp.sys> + 0x4FC3.

   This calculation of the original function address is based on the file classpnp.sys from *Windows XP SP2* (see Figure 8).
3. Next, it detaches DEVICE_OBJECT->AttachedDevice of \Device\HardDiskVolume[N]. This disables the volume level filter driver used by System Restore.
4. It removes the hooks from the IRP_MJ_DEVICE_CONTRL and IRP_MJ_INTERNAL_DEVICE_CONTROL dispatch functions in the atapi.sys driver (the IDE/ATAPI port driver). This disables the filter driver on the IDE/ATAPI port driver.

   It calculates the original function address as follows: it reads from the file atapi.sys, then it searches (in the .INIT section) for the code of the driver object initialization by looking for byte pattern C7 ? 30 ? ? ? ? C7 ? ?, where ? is a wildcard byte (see Figure 9). Assuming the code is found at offset X, Dogrobot reads the data from offset X+0x11 (IdePortDispatch) and

[3] Actually, the second generation sample uses a hard-coded path string, which works on *Windows* Simplified Chinese version only. (This also indicates Dogrobot is designed specifically to target Chinese users.)

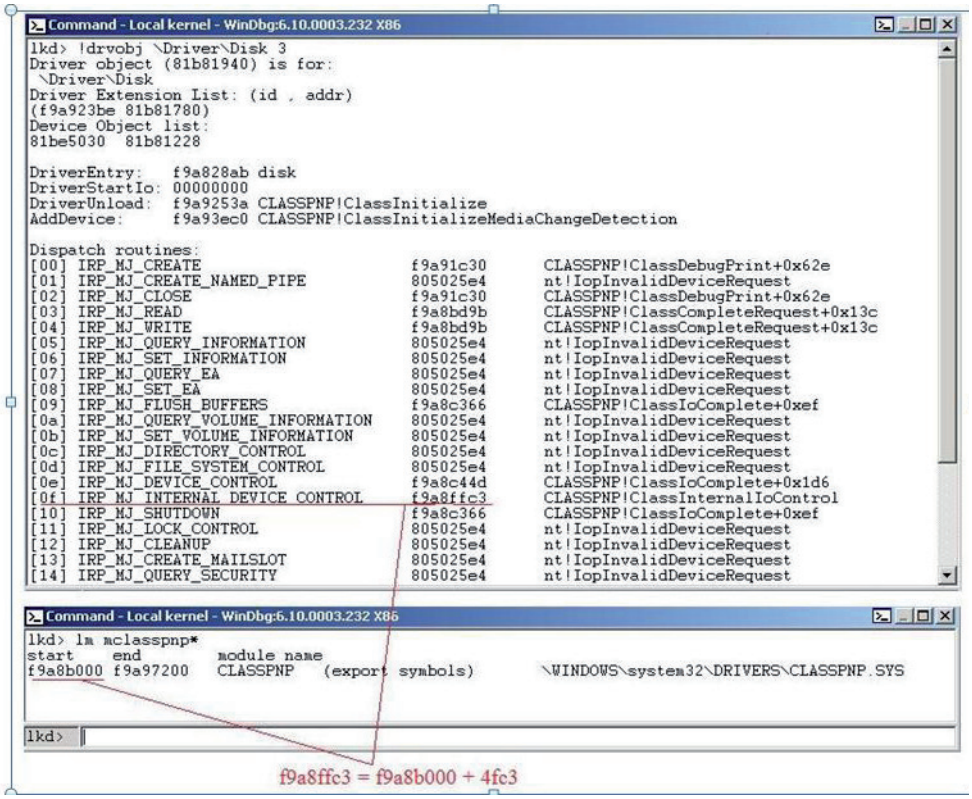

Figure 8: The original dispatch functions for IRP_MJ_INTERNAL_DEVICE_CONTROL on Windows XP SP2.

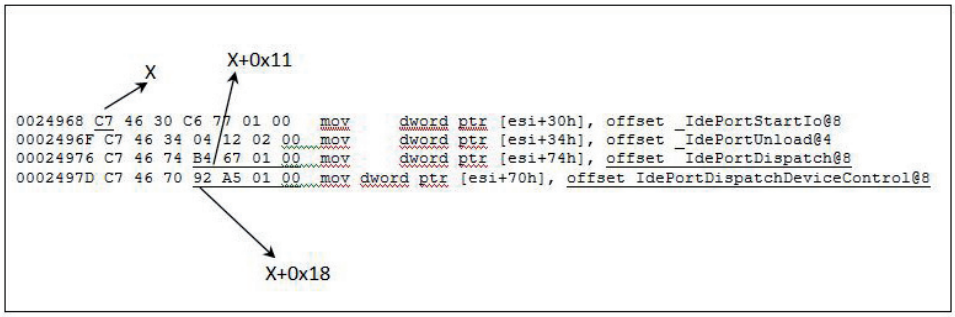

Figure 9: The original IRP_MJ_INTERNAL_DEVICE_CONTROL and IRP_MJ_DEVICE_CONTROL dispatch functions address in atpai.sys.

X+0x18 (IdePortDispatchDeviceIoControl) to restore the original dispatch functions for IRP_MJ_INTERNAL_DEVICE_CONTROL and IRP_MJ_DEVICE_CONTRL respectively. It uses two hard-coded RVA values, 0x67B4 and 0xA592, as the original values if the above-mentioned pattern search fails. This calculation of the original function address is also based on the file atapi.sys from *Windows XP SP2*.

5. It attempts to overwrite the system file conime.exe (Console Input Method Editor) and, if the first attempt fails, the file userinit.bat (this file might be created as a particular Dogrobot immunization solution). Unlike the first generation, the third generation overwrites the target file from the driver rather than from the user-mode application. In order to penetrate the System Restore protection, for any disk level read and write access, it sends IRP_MJ_INTERNAL_DEVICE_CONTROL requests to the device object \Device\Harddisk0\DR0 (it has already removed the hook on IRP_MJ_INTERNAL_DEVICE_CONTROL in step 2).

Fourth generation

The fourth generation of Dogrobot was seen in the wild shortly after the appearance of the third generation, in late June 2008. In this generation, the malware author analysed System Restore patches and enhancements that were designed to defeat previous Dogrobot generations and made the following changes [9]:

1. The fourth generation doesn't call the IoGetDeviceObject Pointer( ) API to get the device object \Device\Harddisk0\DR0 since the author found that some System Restore implementations use inline hooking of this function to prevent Dogrobot retrieving the hard disk device object. Instead, the trojan enumerates each device object in the object directory \Device\Harddisk0 and clears the DeviceObject->AttachedDevice for each one that is found (see Figure 10).
2. It removes any hooks in atapi.sys and the SSDT (System Service Descriptor Table). Unlike the third generation, the fourth generation calculates the original function address in user mode, and when the user-mode application drops the driver, the calculated offset of the original function address is written into the data section of the driver.

Fifth generation

The fifth generation of Dogrobot was noticed in the wild in August 2008. In this generation, Dogrobot uses a new technique, PASS_THROUGH, in order to penetrate through System Restore. *Windows* OS provides three I/O control codes: IOCTL_SCSI_PASS_THROUGH (0x4D004), IOCTL_ATA_PASS_THROUGH (0x4D02C) and IOCTL_IDE_PASS_THROUGH (0x4D028), and user-mode applications can send IRP with these I/O control codes via DeviceIoControl( ) to the disk.sys driver. These IRPs will be

```
0001038E E8 C5 03 00 00  call   ZwOpenDirectoryObject
00010393 8B C0           mov    eax, eax
00010395 0B C0           or     eax, eax
00010397 75 68           jnz    short loc_10401
00010399 8D 7D EC        lea    edi,
[ebp+DirectoryHandle]
0001039C 8D 75 E8        lea    esi, [ebp+Object]
0001039F 6A 00           push   0 ; HandleInformation
000103A1 56              push   esi ; Object
000103A2 6A 00           push   0 ; AccessMode
000103A4 6A 00           push   0 ; ObjectType
000103A6 6A 01           push   1 ; DesiredAccess
000103A8 FF 37           push   dword ptr [edi] ;
Handle
000103AA E8 91 03 00 00  call
ObReferenceObjectByHandle
000103AF 0B C0           or     eax, eax
000103B1 75 47           jnz    short loc_103FA
000103B3 60              pusha
000103B4 8B 1E           mov    ebx, [esi]
000103B6 0B DB           or     ebx, ebx
000103B8 74 38           jz     short loc_103F2
000103BA 33 F6           xor    esi, esi
000103BC EB 2F           jmp    short loc_103ED
000103BE         next_object_entry:
000103BE 8B 0C B3        mov    ecx, [ebx+esi*4]
000103C1 0B C9           or     ecx, ecx
000103C3 74 25           jz     short loc_103EA
000103C5 8B 79 04        mov    edi, [ecx+4]
000103C8 66 8B 07        mov    ax, [edi]
000103CB 66 83 F8 03     cmp    ax, IO_TYPE_DEVICE
000103CF 75 19           jnz    short loc_103EA
000103D1 8B 47 10        mov    eax, [edi+DEVICE_
OBJECT.AttachedDevice]
000103D4 0B C0           or     eax, eax
000103D6 74 12           jz     short loc_103EA
000103D8 A3 10 0A 01 00  mov    dword_10A10, eax
000103DD 89 3D 0C 0A 01 00      mov   dword_10A0C, edi
000103E3 33 C0           xor    eax, eax
000103E5 89 47 10        mov[edi+DEVICE_OBJECT.
AttachedDevice], eax
```

Figure 10: The detaching code used by the fourth generation of Dogrobot samples.

forwarded directly down to the lower driver (e.g. atapi.sys) in order to perform disk read/write or other disk operations [10]. Some System Restore solutions don't intercept the read/write access via PASS_THROUGH and this is exploited by the fifth generation to compromise System Restore. The disassembly of the code used by Dogrobot to write to disk via IOCTL_ATA_PASS_THROUGH is depicted in Figure 11.

Another enhancement that appeared in the fifth generation is the implementation of comprehensive anti-anti-virus features. In user mode, Dogrobot drops the DLL killdll.dll, which contains code to terminate anti-virus processes and to disable anti-virus services. The blacklist which contains the names and IDs of targeted anti-virus processes is passed to the driver

```
mov     [ebp+InputBuffer.CurrentTaskFile+3], al
mov     eax, edx
shr     eax, 10h
mov     [ebp+InputBuffer.CurrentTaskFile+4], al
mov     [ebp+InputBuffer.CurrentTaskFile+2], dl
mov     [ebp+InputBuffer.Length], 28h
mov     [ebp+InputBuffer.TimeOutValue], 0Ah
mov     [ebp+InputBuffer.DataBufferOffset], 2Ch
mov     [ebp+InputBuffer.AtaFlags], 4
mov     edi, 200h
mov     [ebp+InputBuffer.DataTransferLength], edi
mov     [ebp+InputBuffer.CurrentTaskFile], 0
mov     [ebp+InputBuffer.CurrentTaskFile+1], 1
mov     al, [ebx+4]
or      al, 4
shl     al, 4
shr     edx, 18h
or      al, dl
push    0        ; InternalDeviceIoControl
mov     [ebp+InputBuffer.CurrentTaskFile+5], al
lea     eax, [ebp+var_8
push    eax      ; int
lea     eax, [ebp+InputBuffer]
push    eax      ; OutputBuffer
push    22Ch     ; InputBufferLength
push    eax      ; InputBuffer
push    4D02Ch   ; IoControlCode
mov     [ebp+InputBuffer.CurrentTaskFile+6], 30h
push    dword ptr [ebx] ; DeviceObject
call    send_device_io_control
```

Figure 11: The code used by the fifth generation to write disk via IOCTL_ATA_PASS_THROUGH.

component (pcidump.sys). The driver hooks the IRP_MJ_CREATE function in the file system drivers fastfat.sys and ntfs.sys (see Figure 12). The hooked function checks whether the process requesting access matches those in the blacklist, and if so, it returns STATUS_ACCESS_DENIED. This means that any file access attempted by anti-virus software will fail.

WHO LET THE DOG OUT?

Since the appearance of Dogrobot, many hosts have been infected in China. According to statistics published by *Kingsoft*, Dogrobot's infection rate (the percentage of infected hosts among all surveyed hosts) was as high as 6.2% in the first half of 2008 [11]. Such a high rate of infection draws our attention to the propagation method used by this malware. Our analysis of Dogrobot samples did not uncover propagation functionality within their code. Instead, Dogrobot appears to rely on being distributed to target hosts using one of the following methods:

Exploits

The authors/distributors of Dogrobot have spared no effort in attempting to find and exploit vulnerabilities in the *Windows* OS or popular third-party ActiveX controls. They have

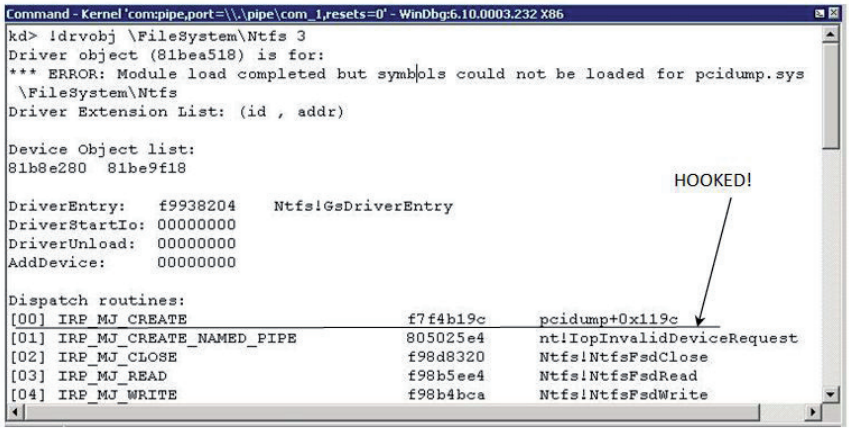

Figure 12: Hooked IRP_MJ_CREATE function in ntfs.sys to disable anti-virus software.

```
<html>
<object classid='clsid:2CACD7BB-1C59-4BBB-8E81-6E83F82C813B' id='target' ></obje
ct>
<script>window.onerror=function(){return true;}</script>
<script language='vbscript'>
arg1="\Program Files\Common Files\uusee\"
arg2="[redacted]"
arg3="[redacted]"
arg4=1
target.Update arg1 ,arg2 ,arg3 ,arg4
</script>
</html>
```

Figure 13: A malicious script used to download Dogrobot via exploits.

developed malicious scripts which download Dogrobot via the exploit of such vulnerabilities. In the wild, we have observed scripts which attempt to exploit vulnerabilities in *Windows* (for example MS08-078), in *RealPlayer* and *WebThunder*. Figure 13 shows a sample which exploits the *UUSee* update ActiveX control in order to download a Dogrobot sample. (*UUSee* is popular online TV software used in China.) The authors/distributors chose popular websites as their targets and hacked these websites in order to upload their malicious scripts. When vulnerable users visited these hacked websites, they were infected by Dogrobot. According to the statistics from *Rising*, in April 2009, several well-known websites including *Eachnet* had been compromised, and there were 860,000 active websites containing malicious scripts [12].

ARP cache poisoning

The second method used for distributing Dogrobot is the so-called ARP cache poisoning. Once an infected host is connected to a network, it can send malicious ARP packets to instruct other machines within the same LAN to download Dogrobot samples.

ARP cache poisoning can be used to hijack the traffic on the LAN (Local Area Network). The mechanism of ARP cache poisoning can be explained as follows:

Let's assume there are three hosts within the same LAN: G is the gateway, V is the victim host, and M is the malicious host (Figure 14). We use IP_G, IP_V, IP_M, ARP_G, ARP_V and ARP_M to denote their IP addresses and ARP addresses. In the normal scenario, when V is communicating with gateway G, V will broadcast an ARP query, 'who has the IP address of IP_G?', and G will reply with 'I have the IP address of IP_G, and my ARP is ARP_G'. So V will save IP_G/ARP_G in its ARP cache. And G will do the same thing for V in its ARP cache. When M is attacking, M sends out two poison packets [13]:

1. It sends an ARP reply to V, pretending to be the gateway for V: 'I have the IP address of IP_G, and my ARP is ARP_M', hence V will cache the wrong ARP binding IP_G/ARP_M, and all the outbound traffic sent to gateway IP address (IP_G) will be sent to M.
2. Then it sends another ARP reply to G, pretending to be host V: 'I have the IP address of IP_V, and my ARP is ARP_M'. Accordingly, all the traffic sent to V (IP_V) will be rerouted to M, i.e. all inbound traffic to V will be sent to M.

So all the inbound and outbound traffic is finally hijacked by M, and M can alter the data and forward it to the original destination (outbound to G, and inbound to V), thus performing a 'man-in-the-middle' attack [13].

A Dogrobot-infected machine downloads a trojan which has an ARP spoof payload and acts as an ARP cache poisoning attacker: it hijacks all the HTTP (TCP port 80) traffic to itself, and may alter web page content (sent from a web server as HTTP traffic), by adding an IFRAME which points to the malicious script that downloads a Dogrobot sample. Hence, once one host gets infected, the other hosts in the same LAN are likely to be comprised as well. This propagation method is particularly successful in Internet cafés, where a number of computers are connected within the same LAN.

THE DRINKER'S HEART IS NOT IN THE CUP

According to [14], 'The hard drive recovery card was introduced over 20 years ago', while the first malware that targeted System Restore was observed in the wild in late 2007. The reason for compromising System Restore protection is clear when we consider the fact that many Internet cafés in China use System Restore in order to keep the integrity of their machines intact. Indeed, Dogrobot has been designed deliberately to target Internet cafés in China. So what is the significance of breaking System Restore and what is the main goal of this malware? Further analysis of Dogrobot indicates that the main payload of this malware is downloading other malware to the affected machines. While tracking other malware downloaded by Dogrobot variants, we found that most of them were related to various online game password-stealing applications, like Zuten [15], Frethog [16], Tilcun [17] or Lolyda [18].

In 2008, we discussed various online game targeting malware, which are often subject to trade on the black market [4]. Now, it is clear that the birth and development of Dogrobot has been driven by the operation of the black market trade in compromised online game accounts. Dogrobot is a part of the larger picture and is a component that is utilized to ensure the ongoing survival of other malware.

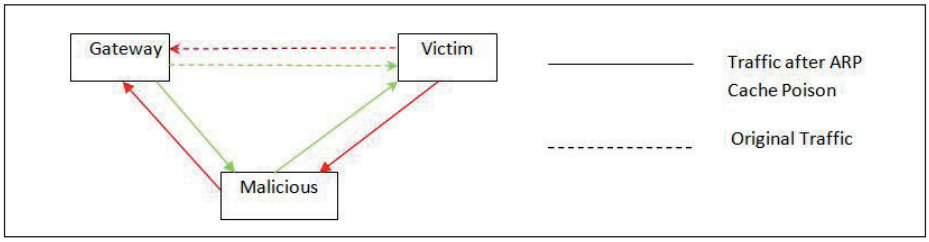

Figure 14: The mechanism of an ARP cache poisoning attack.

CONCLUSION

Dogrobot is the first malware observed in the wild that operates at a low level and targets System Restore. In lab testing, we found that its code appears not to have been developed in a meticulous manner – there are a number of bugs in the malware which may prevent it from functioning properly, or cause the affected system to crash. However, we still believe the author has extensive knowledge of System Restore and rootkit technologies. The evolution of Dogrobot also shows that the malware author has been closely monitoring progress in System Restore and anti-virus technologies and has been making improvements and modifications in response to changes in those security products.

So far, we have seen Dogrobot evolve through five generations. However, it appears that its malicious work will continue and we should expect more of the same with continued development and refinement.

Dogrobot poses a challenge not only to the anti-virus industry, but also to System Restore vendors and IT administrators. Vendors of system maintenance products have started developing new diskless workstation systems. In such systems, workstations have no hard disk and the information is stored on the server; this changes the way the whole system is protected and it also changes the playground for malware and malware authors.

REFERENCES

[1] Product details for HD Recovery Card V9.2. http://www.computersentry.net/product_detail.asp?Product_ID=474.

[2] Drive Vaccine. http://www.horizondatasys.com/231846.ihtml.

[3] http://news.ccw.com.cn/soft/htm2008/20080805_478450.shtml.

[4] Feng, C. Playing with shadows – exposing the black market for online game password theft. Proceedings of the 18th Virus Bulletin International Conference, 2008.

[5] http://www.horizondatasys.com/311847.ihtml.

[6] https://www.eviloctal.com/thread-33489-1-1.html.

[7] FileDisk. http://www.acc.umu.se/~bosse/.

[8] http://bbs.duba.net/thread-21943960-1-4.html.

[9] https://forum.eviloctal.com/thread-33451-1-5.html.

[10] Advanced Bootkit: Tophet. XCON Conference 2008.

[11] http://www.duba.net/zt/08report/.

[12] http://www.022net.com/2009/4-23/47217533255829.html.

[13] ARP cache poisoning. http://www.grc.com/nat/arp.htm.

[14] Hard drive recovery cards – the evolution. http://www.harddriverecoverycards.com/jungsoft_hdd_sheriff.php.

[15] Microsoft Malware Protection Center. http://www.microsoft.com/security/portal/Entry.aspx?Name=PWS:Win32/Zuten.

[16] Microsoft Malware Protection Center. http://www.microsoft.com/security/portal/Entry.aspx?Name=PWS:Win32/Frethog.

[17] Microsoft Malware Protection Center. http://www.microsoft.com/security/portal/Entry.aspx?Name=Trojan:Win32/Tilcun.

[18] Microsoft Malware Protection Center. http://www.microsoft.com/security/portal/Entry.aspx?Name=Win32%2fLolyda.

[19] Russinovich, M.E.; Solomon D.A. Microsoft Windows Internals (4th Edition).

APPENDIX

The taxonomy of the *Windows* File System Driver is depicted in Figure 15 [19] (see next page).

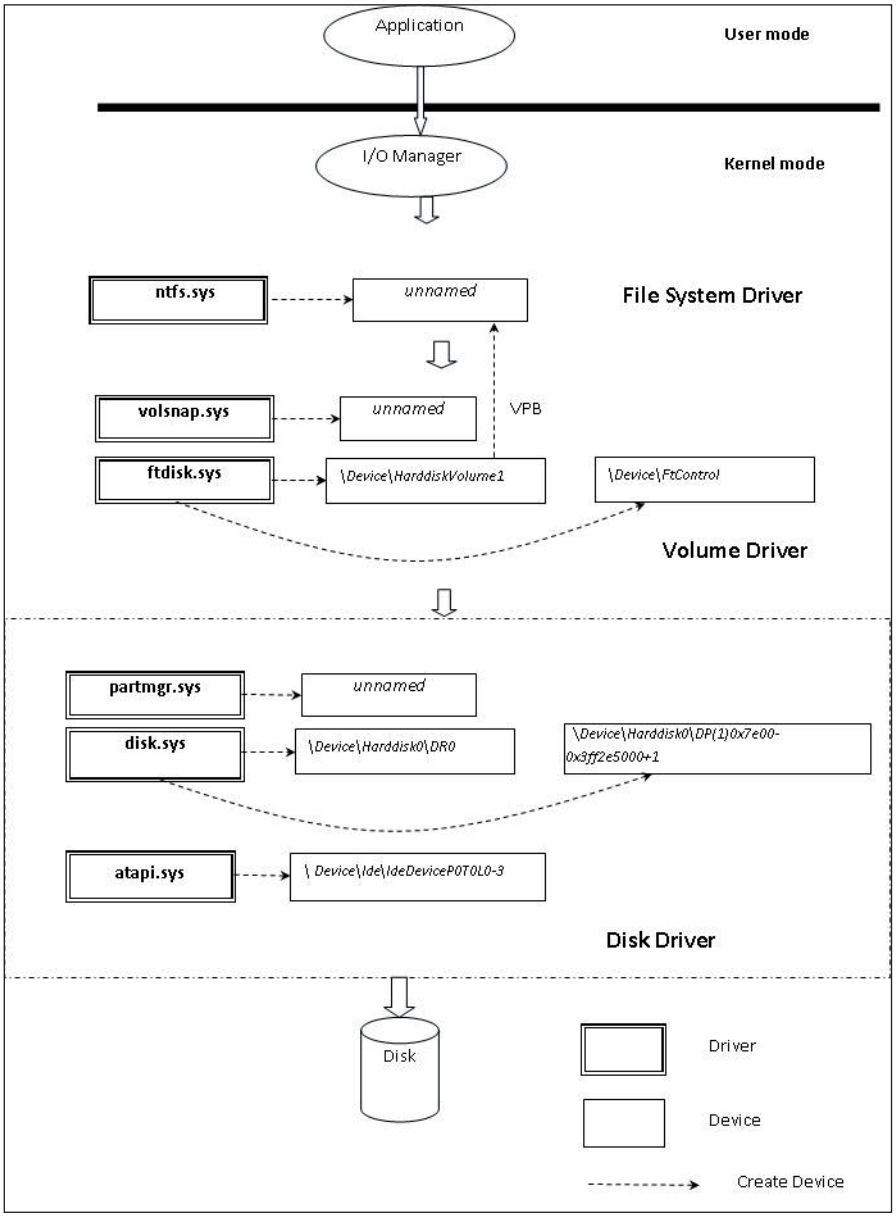

Figure 15: The taxonomy of Windows file system driver.