botnet is owned by a Russian
botnet is attacking a forum that participates in illegal activities
The botnet itself is doing what can be seen as overall good. (utilitarianism). the zombies are losing some of their bandwidth and computation speed but their information of private citizens is being defended.

[With no explicit obligation to my firm, under utilitarianism]
I would tell my friend and let them choose what action to take on their machine, observe the botnet activities, then finally seize the botnet when the time seems right. The right time may come later because while they are DDosing the sharing forum, we can properly formulate a plan of action towards both parties.

I go with the observe and take plan approach because I view breech of private information a more severe violation of freedoms than wrangling zombies as resources. Law also has not caught up to cyber activities and has not matured to guide certain online activities. The practice of violating privacy however, has been around much longer and can have more severe consequences; ie. revealing private identities to compromise opsec, identity fraud etc.

The botnet is doing more good than bad and should be allowed to exist until the forum is dealt with. Following that, dismantle the botnet enough is learnt from it.
This is assuming of course that the resources used by each zombie is not majorly affecting the machine's owners.

[if we are able to get a backdoor into the botnet]
If we could infiltrate the botmaster's system, it might be beneficial to allow those activities to continue because we have a kill switch. And then only use the switch it if activities no longer follow the "thieves code" where only illegal/dishonest systems are attacked. (*dishonest is subjective, as many business do less than enviable actions but shouldn't be targeted)

[with explicit obligation to my firm with NDA]
Under any restriction where information is protected under NDA I would refrain from letting my friend know because I'm binded to a certain behaviour and would prefer to incur as little penalty to myself as possible (egoism). At very worst my friend is then just a sucker who is stuck with slow internet at times.

stuxnet analysis (30 points).

1. Provide an explanation of its three components (worm, link file, root kit) and how they interact with each other (10 points).

worm:

To spread the via usb or network. (more on this in question 2)

1. by usb: waiting on “WM_DEVICECHANGE” windows message to notify, and then if the inserted drive it qualifies for action it will remove or write 4 shortcuts and 2 DLL files. As part of the LNK vulnerability, these shortcuts are auto run when the folder is viewed.
2. by network:
    a. execute shell code on a connected system by using a special RPC request
    b. using a 0-day vulnerability. using a windows API, creates a file in a directory, then using “GetSpoolFileHandle” to get the file handle to modify the file (read/write to target machine).
    2 files are copied onto the target: System32\winsta.exe (stuxnet dropper), System32\wbem/mof/sysnullevnt.mof (managed object format file. which under some conditions executes the stuxnet dropper)

rootkit:

The purpose of the rootkit is to get admin rights to modify files and hide stuxnet files. (is WTR4141.tmp part of the rootkit? might not be but i guess it belongs in this section.)

~WTR4141.tmp and ~WTR4132.tmp work to hide files. 4141 is up first and hides files, then loads 4132 and pass control.

MRxNet adds itself to driver chains for ntfs, fastfat, and cdfs, and receives I/O request packets ISPs first and modifies the packets before allowing other drivers to receive them. Entries that seem like stuxnet files are deleted.

link file:

The lnk file is auto run when the folder is viewed, and runs ~WTR4141.tmp

Working together:

- The link file is exactly 4171 bytes, .lnk files with size of 4171 bytes are hidden by the rootkit

- ~WTR4141.tmp hides ~WTR4131.tmp and identifies it by ~WTRxxxx.tmp where xxxx mod 10 = 0
- the worm copies the lnk file and dll files for rootkit

2. Provide an explanation of the vulnerabilities it took advantage of (6 points).
Four 0-day vulnerabilities in and 1 previously known vulnerability known as conficker in the windows OS.

Two of the 0-day vulnerabilities in were used to escalate privileges to admin level:

Win32K.sys Keyboard layout Vulnerability

A keyboard layout file is loaded which allows arbitrary code to be executed with system privileges.

Windows Task Scheduler Vulnerability

A user can leverage the task scheduler to execute files they do not have permissions to.

The purpose of both the above vulnerabilities is to create a new process/task for the worm to run in.

Conficker (not zero day)

An RPC request is used to create a buffer overflow allowing shell code to be executed. This allows stuxnet to spread via a connected network.

Windows Print Spooler Service Vulnerability

A guest user account (ie. low privilege) communicate with a target machine via a shared printer allowing it to write to the machine's system directory using a windows API and another API to get a file handle on the target machine.

Windows Shell LNK Vulnerability

Stuxnet waits on the WOM_DEVICECHANGE windows message for when a new drive is detected, and writes 4 shortcuts to .lnk file and 2 DLL files. The 4 shortcuts are for compatability with multiple windows versions.

http://www.codeproject.com/Articles/246545/Stuxnet-Malware-Analysis-Paper#ch4.4.1

https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf
http://www.welivesecurity.com/2010/10/15/win32k-sys-about-the-patched-stuxnet-exploit/
https://technet.microsoft.com/en-ca/library/security/ms15-028.aspx

3. Explain in your own words, what appears to be the main target of Stuxnet? What impact could Stuxnet have in the targets infrastructure? (3 points).
According to [source] the main target were centrifuges being used to enrich uranium. The worm would propagate to many machines but only take extra action in a certain case:

type of PLC used
what the PLC is doing - hooked up to a centrifuge
how fast the centrifuge is spinning
then mess with the centrifuge RPM at certain intervals points for specific durations.

Because the time that stuxnet would change centrifuge RPM was relatively short and the sleep time was long, it might be difficult for people to observe a centrifuge behaving abnormally, and if they did, determine why it happened and returned to normal operation.

4. Compare it with related malware: Duqu and Flame (5 points).
Duqu was the next big virus to come out after stuxnet. If the certain PLC used in the centrifuges was found, it would monitor the system by taking screenshots and keylog.
Flame was Duqu's upgrade / big brother, it would also record skype conversations, webcams, audio.

5. In your opinion, is Stuxnet a game-changer malware as the media puts it? Justify your answer (6 points).
I would side with that stuxnet was a game changer for a few reasons.

- the size of the program, complexity, and targeting suggests that it's creation was state sponsored. Cyber warfare. The targeted machines were very specific, exact PLCs and other hardware/software had to be known, and the behaviour of the centrifuges was further monitored before interfering. Access

to that knowledge and the know how to create stuxnet which spans so many systems for vulnerabilities is not likely to be found by a few people working in their spare time. Vulnerabilities could have been sold for potentially 6-digit dollar values for personal gain.

- media and larger crowds deem it as a game changer, and that in itself causes waves

Runnable program:

file 1 / first guards that return 0x1-0x3:

The first exit possibility that returns 0x1 checks that the call to get_datafile() did not return an error code.

The next exit possibility “0x2” does something with checking that the lower byte of variable at esp+0x28 is not equal to lower byte of (0x8048a0e). Which with some digging says: check if esp+0x28 is not 0x30 == ascii 0. Not sure what having a zero here COULD have done, but easy win i guess.

Exit possibility “0x3” checks that the contents of file 1 does not have some sort of negative bytes.

file2:

Exit possibility 0x5 compares contents of file 2 with 0xef2a, which means the contents of the file should contain 0x2aef to proceed. I used a C program to write these bytes into the file.

file 3:

Exit possibility 0x6 has multiple checks.
The contents of file 3 get read and then placed onto the stack when strncpy() gets called which actually overwrites the memory locations (local vars) that were checked by guard 1 and 2. Using a token of “AAAABBBBCCCC…” which changes every 4 bytes i was able to achieve this on the stack:

memory location	**value**		**New value to pass compares**
	AAAA		
	BBBB		
	CCCC		

	DDDD		
	EEEE		
	FFFF		
	GGGG		
	HHHH		
esp + 0x38	IIII	third compare	UUUU
esp + 0x3c	JJJJ	second compare	DDDD
	KKKK		
esp + 0x44	LLLL	first compare	3333

UUUU = 0x55555555
DDDD = 0x44444444
3333 = 0x33333333
These three values pass the compares and allows program to proceed.

File 4:
Exit possibility 0x7 occurs if the call made to _____() returns without error, this function doesn't actually do anything other than call strlen() and strncpy(), the same function used to overflow with file 3. Because _____() does not on any of the execution paths return an error code we should look to exploit strncpy() again, but this time seek to overwrite the saved ebp.

Using the same scheme as used for file 3, I input more aaaabbbbccccdddd…. and got an error saying "Cannot access memory at address 0x67676767". 0x67 == g.
Truncate file 4 to remove everything including gggg and afterwards, then use a c program to append the byte value 0xad880408. So the saved ebp is changed to 0x080488ad before returning from _____(). Exit 0x7 is bypassed and we jump to the hello world area.