

1) *Is this malware packed? If so, with which packer?*

It is packed with Themida, produced by Oreans (<http://www.oreans.com/themida.php>), which is a very hard to break protector. It includes optionally detection for virtual machines (VirtualPC and VMware), a feature which is activated in this executable. Because of this analysis is only possible with an other virtual machine (if you don't want to sacrifice a real one :)), my personal choice being Qemu (<http://fabrice.bellard.free.fr/qemu/>).

During the process of unpacking the executable dumps a driver in the system32\drivers directory with the name oreans32.sys (size: 33952 bytes, MD5: aad837bf3b475092fd515cd0842334e9). Although this is detected by CAT-QuickHeal as Rootkit.Agent.ad, is not malicious file, but a part of the protector itself. Still, probably most people would like to remove this file too, which can be done by entering `net stop oreans32` at the command prompt and then deleting the file itself. Given the way this file was packed, we can say for certain that it was first run by a program having the load driver privilege (usually this is attributed to members of the Administrators group only). On subsequent runs however it wasn't necessary for it to have this privilege since the driver was already installed and running. This is an other reason to delete the driver.

2) *What is the purpose of this malware?*

It has a single purpose: to download and execute any file the master controller instructs it to.

3) *Does it connect to a remote server? With which purpose?*

It connects to `uk.undernet.org` and joins the channel `#secretcow` with the password `werule` which acts as a command and control center for it (the channel has been shut down). The nickname of the user who controls this bot is Daddy.

4) *Which channels does it connects to?*

`#secretcow`. It also sends status messages to the `#yousawthatss` channel (like `:Downloading.` and `:Executing.`)

5) *Can you get any passwords related to this malware? (Not the infected password) :*

The channel password for `#secretcow` is / was `werule`

6) *Which capabilities does this malware have?*

To download and execute files from a given url. These files are downloaded to `C:\dl.exe` and executed from there. This is an other indication for the fact that the malware was intended to run with high privileges (because low privileged users do not have write access to `c:\`). To reconstruct the files which were downloaded and executed (if any), there are two possibilities: looking at the Internet Explorer cache on the infected machine. This works because the malware uses the `URLDownloadToFile` API, which is part of Internet Explorer and stores the downloaded files in the same cache as it. An other possibility would be to look at the proxy logs if the computer was behind one. Because the malware uses a function which is part of Internet Explorer, it uses the same proxy settings as IE.

7) *What is the hidden message? (if there is any...) :*

`no security without knowledge. No knowledge without research. Be a good guy! It is worthwhile.`

For analysis I ran the program in Qemu for a couple of seconds and then did a full memory dump of the malware using the small program shown in the appendix. I imported the relevant sections (`dump_00400000.bin`, `dump_00401000.bin` and `dump_00404000.bin`) in IDA (when importing you must take care to mark the sections as 32 bit code). Unfortunately the protector obfuscates the IAT, so the analysis was done mostly by looking at the strings and the method calls to the library functions (it was written in Delphi and IDAs FLIRT signatures managed to identify a couple of library routines).

## Appendix – Source code for the memory dumper Use Turbo Delphi or FreePascal to compile.

```
program dump_mem;

{$APPTYPE CONSOLE}

uses
  SysUtils,
  Windows;

var
  hProcess: THandle;
  l: integer;
  buffer: pointer;
  numRead: Cardinal;
  memStart: pointer;
  memInfo: MEMORY_BASIC_INFORMATION;

procedure PutFile(FileName: String; FileDataPtr: Pointer; FileSize: Cardinal);
var
  hFile: THandle;
  Written: Cardinal;
begin
  hFile := CreateFile(PChar(FileName), GENERIC_WRITE, 0, nil, CREATE_NEW, FILE_ATTRIBUTE_NORMAL, 0);
  if hFile=INVALID_HANDLE_VALUE then begin
    WriteLn('Failed to open output file!');
    exit;
  end;

  WriteFile(hFile, FileDataPtr^, FileSize, Written, nil);

  If Written <> FileSize then begin
    WriteLn('Failed to write output file!');
    exit;
  end;

  CloseHandle(hFile);
end;

var
  pid: Integer;
begin
  pid := StrToInt(ParamStr(1));

  hProcess := OpenProcess(PROCESS_ALL_ACCESS, false, pid);
  if hProcess <> 0 then begin
    memStart := nil;
    l := VirtualQueryEx(hProcess, memStart, memInfo, SizeOf(MEMORY_BASIC_INFORMATION));
    while l = SizeOf(MEMORY_BASIC_INFORMATION) do begin
      if memInfo.State = MEM_COMMIT then begin
        GetMem(buffer, memInfo.RegionSize);
        if ReadProcessMemory(hProcess, memInfo.BaseAddress, buffer, memInfo.RegionSize, numRead) then begin
          PutFile('dump_' + IntToHex(Integer(memInfo.BaseAddress), 8) + '.bin', buffer, memInfo.RegionSize);
          end;
          FreeMem(buffer);
        end;
        integer(memstart) := integer(meminfo.BaseAddress) + meminfo.regionsize;
        l := VirtualQueryEx(hProcess, memStart, memInfo, SizeOf(MEMORY_BASIC_INFORMATION));
      end;
    end;
    CloseHandle(hProcess);
  end.
end.
```