**Lab 5**

只用IDA Pro分析在文件Lab05-01.dll中发现的恶意代码。

**Questions**

1. DllMain的地址是什么?
2. 使用Imports窗口并浏览到gethostbyname, 导入函数定位到什么地址?
3. 有多少函数调用了gethostbyname?
4. 将精力集中在位于0x10001757处的对gethostbyname的调用，你能找出哪个DNS请求将被触发吗?
5. IDA Pro识别了在0x10001656处的子过程中的多少个局部变量?
6. IDA Pro识别了在0x10001656处的子过程中的多少个参数?
7. 使用Strings窗口，在反汇编中定位字符串\cmd.exe /c。它位于哪?
8. 在引用\cmd.exe /c的代码所在区域发生了什么?
9. 在同样的区域，在0x100101C8处，看起来好像dword\_1008E5C4是一个全局变量，它帮助决定走哪条路径。那恶意代码是如何设置dword\_1008E5C4的呢?（提示：使用dword\_1008E5C4的交叉引用）
10. 在位于0x1000FF58处的子过程中的几百行指令中，有一系列使用memcmp来比较字符串的指令。如果对rotbotwork的字符串比较是成功的（memcmp返回0），会发生什么?
11. PSLIST导出函数做了什么?
12. 使用图模式来绘制出对sub\_10004E79的交叉引用图。当进入这个函数时，哪个API函数可能被调用? 仅仅基于这些API函数，你会如何重命名这个函数?
13. DllMain直接调用了多少个Windows API? 多少个在深度为2时被调用?
14. 在0x10001358处，有一个对Sleep（一个使用包含要睡眠的毫秒数的参数的API函数）的调用。顺着代码向后看，如果这段代码执行，这个程序会睡眠多久?
15. 在0x10001701处是一个对socket的调用。它的3个参数是什么?
16. 使用MSDN页面的socket和IDA Pro中的命名符号常量，你能使参数更加有意义吗? 在你应用了修改以后，参数是什么?
17. 搜索in指令（opcode 0xED）的使用。这个指令和一个魔术字符串VMXh用来进行VMware检测。这在这个恶意代码中被使用了吗? 使用对执行in指令函数的交叉引用，能发现进一步检测VMware的证据吗?
18. 将你的光标跳转到0x1001D988处，你发现了什么?
19. 如果你安装了IDA Python插件（包括IDA Pro的商业版本的插件），运行Lab05-01.py，一个本书中随恶意代码提供的IDA Pro Python脚本，（确定光标是在0x1001D988处）在你运行这个脚本后发生了什么?
20. 将光标放在同一位置，你如何将这个数据转成一个单一的ASCII字符串?
21. 使用一个文本编辑器打开这个脚本。它是如何工作的?