

# Kernel-22

## A Framework for Creating Analysis Tools

Mike McCarl

ICSALabs

[mmccarl@icsalabs.com](mailto:mmccarl@icsalabs.com)



Copyright 2009 Cybertrust. All Rights Reserved.

## What is Kernel-22?

- **A source code framework that can be used to produce alternate kernel32.dll modules**
  - Produce a kernel32.dll module that can be used by any Win32 application
  - Developers can easily augment the functionality of kernel32.dll by adding custom code
  - Functionality of “the real” kernel32.dll is retained
- **Consists of 4 source files**
  - 2 of which can be generated from the output of the dumpbin utility
  - 1 header file
  - 1 containing 2 simple functions (including DllMain)
- **Without modification, the framework produces a kernel32.dll module that has no apparent effect**

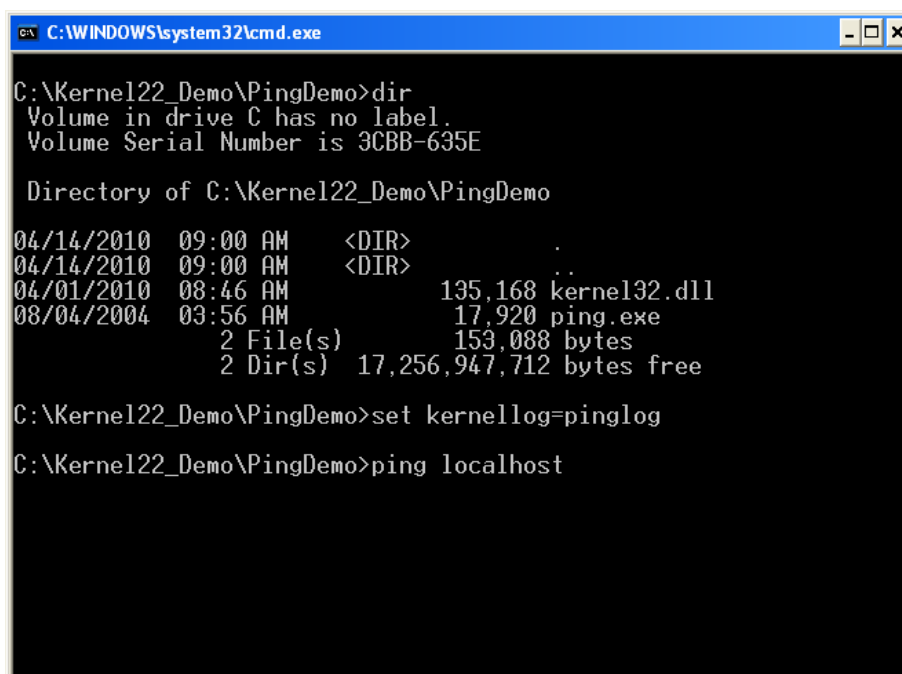
## What makes it special?

- **There are other tools/programs/techniques that get between an application and kernel32.dll**
  - Some use hooks
  - Some inject code
  
- **Kernel-22 avoids complex coding to get into the system**
  
- **It is “special” simply because it uses “ordinary” means to operate**
  
- **For complete details, Kernel-22.doc is available for download**

## Why use it?

- **Consider a reverse-engineering debug session. You might:**
  - Do static analysis to identify the malware's dependencies and create a debugging strategy
  - Set breakpoints on functions in kernel32.dll to examine parameters or other application states
  - Alter return codes or data to simulate certain conditions
- **Suppose that instead you could run the program and produce a log of the kernel32 functions called**
  - A more complete list of the functions called would be produced (especially if the malware was packed)
  - The sequence of called functions is much clearer
  - Faster
  - Could be automated!

## Demonstration: Logging Kernel32 Calls



```
C:\WINDOWS\system32\cmd.exe

C:\Kernel22_Demo\PingDemo>dir
Volume in drive C has no label.
Volume Serial Number is 3CBB-695E

Directory of C:\Kernel22_Demo\PingDemo

04/14/2010  09:00 AM    <DIR>          .
04/14/2010  09:00 AM    <DIR>          ..
04/01/2010  08:46 AM                135,168 kernel32.dll
08/04/2004  03:56 AM                17,920 ping.exe
                2 File(s)      153,088 bytes
                2 Dir(s)  17,256,947,712 bytes free

C:\Kernel22_Demo\PingDemo>set kernellog=pinglog
C:\Kernel22_Demo\PingDemo>ping localhost
```

## Demonstration: Logging Kernel32 Calls

```
C:\WINDOWS\system32\cmd.exe
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Kernel22_Demo\PingDemo>dir
Volume in drive C has no label.
Volume Serial Number is 3CBB-635E

Directory of C:\Kernel22_Demo\PingDemo

04/14/2010  08:52 AM    <DIR>          .
04/14/2010  08:52 AM    <DIR>          ..
04/01/2010  08:46 AM             135,168 kernel32.dll
08/04/2004  03:56 AM             17,920 ping.exe
04/14/2010  08:52 AM           404,069 pinglog.000002cc
                3 File(s)      557,157 bytes
                2 Dir(s)  17,256,742,912 bytes free

C:\Kernel22_Demo\PingDemo>
```



## How It's Done

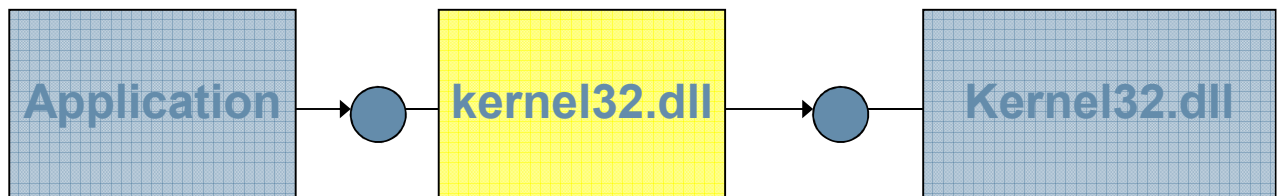
- **Registry Key \HKLM\SYSTEM\CurrentControlSet\Control\Session Manger**
  - Value ExcludeFromKnownDlls (REG\_MULTI\_SZ): Append "Kernel32.dll"
  - Reboot after change
- **Loader will now use search path rules to locate kernel32.dll**
- **Put the new kernel32.dll and the program to be monitored in the same directory**
- **Run the program**



## WAIT A MINUTE!

- If the “local” kernel32.dll is loaded, how does it create a log file?
- How does it implement all those functions that were called?
- Answer: It doesn't. The “real” kernel32.dll does.

## Apparent Module Structure



## The Mechanics of DLLs

- **DLLs may be made available to a process in 2 ways:**
  - Implicitly: A module of the process contains an Import Table which specifies the names of required DLLs and the functions used. The operating system reads this table at load time and makes the specified DLLs available before execution begins.
  - Explicitly: The process itself calls for the loading of a DLL by calling one of the LoadLibrary functions. The load of the library occurs *after* execution of the process begins.
- **Addresses to imported functions of an implicitly loaded DLL are resolved at load time**
  - References to functions are initially resolved at link time to other references in the import table. At load time the loader simply has to fill in the “real” addresses.
  - To initially resolve the import addresses, the linker requires a “.lib” file. The lib file specifies the name of the dll that needs to be loaded as well as the names of the functions that are available.
- **Addresses to functions in a DLL loaded explicitly can be determined by calling GetProcAddress for the desired function.**

## Explicit Load Option

- **If we try to load the real kernel32.dll explicitly, we encounter a dilemma immediately**
  - The LoadLibrary must be called to load kernel32.dll
  - The kernel32.dll we are trying to build also exports a function called LoadLibrary. This function doesn't actually do anything. It will eventually try to call LoadLibrary again, which makes it infinitely recursive.
  - Even if you could avoid the infinite loop (which can be done), you still need to have the real kernel32 loaded in order to call the LoadLibrary function to load it.
  
- **Conclusion: Explicit loading of the real kernel32.dll won't work.**

## Implicit Load Option

- The loader reads the import table of a module to determine which other modules to load
- The Dll name supplied in the import table does NOT contain the full path of the Dll
- Using “search path” rules, the loader first looks in the directory from which the module was loaded
- If our kernel32 requires a module named kernel32, the loader will look in the directory from which our kernel32 was loaded, and load it again!
- Implicit loading of the real kernel32.dll won't work
- Catch-22!



# What if...?



**copy kernel32.dll kernelkernel.dll**

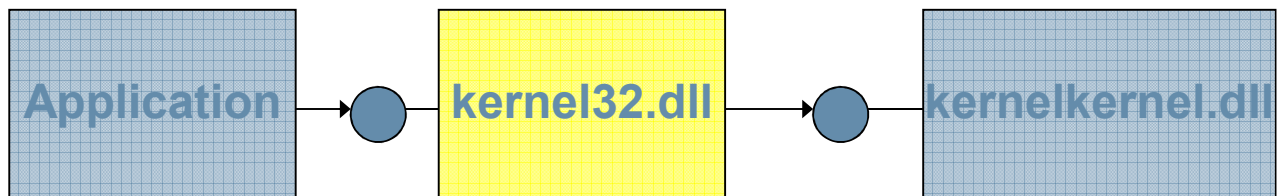
## Resolving the Catch

- There's nothing that says that kernel32.dll must be named kernel32.dll.
- It is simply a convention that all programs that wish to use that particular set of functions know to use kernel32.dll
- If we copy kernel32.dll to another file name, we will have a module that still provides the same functions, but the loader won't confuse it with *our* kernel32.dll





## Actual Module Structure



## The Next Problem...

- How to make our kernel32.dll dependent on kernelkernel.dll



## The Source of a Dependency

- **The referenced dll name comes from the lib file that the linker uses to initially build the import table.**
  - The .lib file gets it from the “LIBRARY” statement in the .def file used to build the dll
  
- **How can we change it?**
  - A .lib file is a binary file, so using a hex editor might work, but this isn’t very robust and may be risky.
  - A better way is to build our own kernelkernel.lib from scratch.

## Building KernelKernel.lib

- **.Lib files are created in conjunction with .Dll files, so we need a project to create kernelkernel.dll**
- **The only functions that need to be implemented are those that our kernel32.dll will call**
  - GetModuleHandle
  - GetProcAddress
- **Prototypes for these functions are available in the PlatformSDK header files**
- **All we want is the .lib file, the dll file created is superfluous. Therefore, the function bodies we create can be empty.**
- **When the compile/link is complete, throw away the kernelkernel.dll and link the .lib file with our kernel32.**
- **As more functionality is desired, add other functions to kernelkernel or use GetProcAddress to locate them in kernelkernel.dll.**

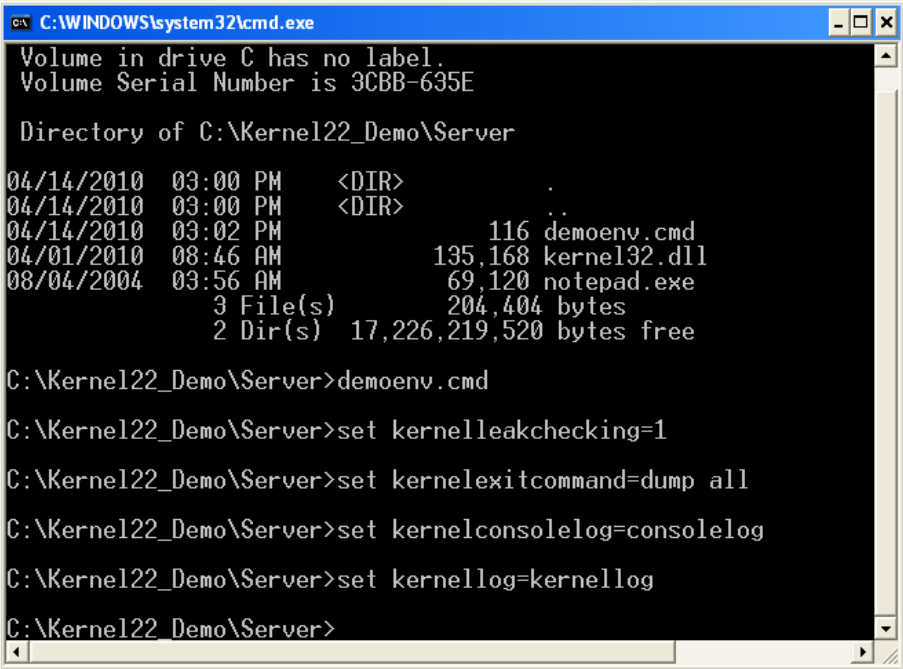
## Potential Uses

- Detect memory leaks or code injections
- Unpacking
- Simulations
- Protect files

## Demonstration: Memory Leak Detection

- **Framework is modified to add additional processing to the memory allocation/release functions**
  - GlobalAlloc, GlobalReAlloc, GlobalFree, GlobalDiscard
  - HeapAlloc, HeapRealloc, HeapFree
  - FormatMessageA, FormatMessageW
  - LocalAlloc, LocalDiscard, LocalReAlloc, LocalFree
  - TlsAlloc, TlsSetValue, TlsFree
  - VirtualAlloc, VirtualAllocEx, VirtualFree, VirtualFreeEx
  
- **When memory is allocated, add an entry to a list.**
  
- **When memory is freed, remove the corresponding entry from the list**
  
- **At the end of the program, anything still in the list is leaked memory.**

## Demonstration: Memory Leak Detection



```
C:\WINDOWS\system32\cmd.exe
Volume in drive C has no label.
Volume Serial Number is 3CBB-635E

Directory of C:\Kernel22_Demo\Server

04/14/2010  03:00 PM    <DIR>          .
04/14/2010  03:00 PM    <DIR>          ..
04/14/2010  03:02 PM                116 demoenv.cmd
04/01/2010  08:46 AM           135,168 kernel32.dll
08/04/2004  03:56 AM             69,120 notepad.exe
              3 File(s)        204,404 bytes
              2 Dir(s)  17,226,219,520 bytes free

C:\Kernel22_Demo\Server>demoenv.cmd

C:\Kernel22_Demo\Server>set kernelleakchecking=1

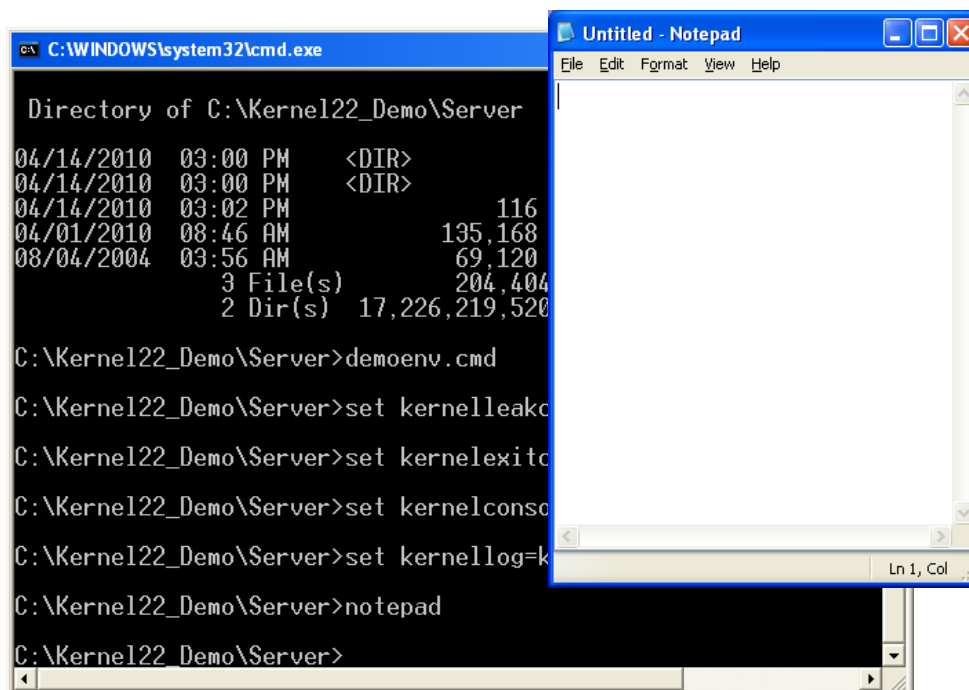
C:\Kernel22_Demo\Server>set kernelexitcommand=dump all

C:\Kernel22_Demo\Server>set kernelconsolelog=consolelog

C:\Kernel22_Demo\Server>set kernellog=kernellog

C:\Kernel22_Demo\Server>
```

## Demonstration: Memory Leak Detection



```
C:\WINDOWS\system32\cmd.exe

Directory of C:\Kernel22_Demo\Server

04/14/2010  03:00 PM    <DIR>
04/14/2010  03:00 PM    <DIR>
04/14/2010  03:02 PM                116
04/01/2010  08:46 AM            135,168
08/04/2004  03:56 AM             69,120
               3 File(s)            204,404
               2 Dir(s)       17,226,219,520

C:\Kernel22_Demo\Server>demoenv.cmd
C:\Kernel22_Demo\Server>set kernelleak=1
C:\Kernel22_Demo\Server>set kernelexit=1
C:\Kernel22_Demo\Server>set kernelconsole=1
C:\Kernel22_Demo\Server>set kernellog=kerneldump.log
C:\Kernel22_Demo\Server>notepad
C:\Kernel22_Demo\Server>
```



## Demonstration: Memory Leak Detection

```
C:\WINDOWS\system32\cmd.exe - KernelConsole.exe \\.\pipe\kernel22.000008d4

C:\Kernel122_Demo\Client>dir ..\Server
Volume in drive C has no label.
Volume Serial Number is 3CBB-635E

Directory of C:\Kernel122_Demo\Server

04/14/2010  03:13 PM    <DIR>          .
04/14/2010  03:13 PM    <DIR>          ..
04/14/2010  03:13 PM                0 consolelog.000008d4
04/14/2010  03:02 PM                116 demoenv.cmd
04/01/2010  08:46 AM            135,168 kernel32.dll
04/14/2010  03:13 PM            538,919 kernellog.000008d4
08/04/2004  03:56 AM             69,120 notepad.exe
           5 File(s)              743,323 bytes
           2 Dir(s)  17,225,555,968 bytes free

C:\Kernel122_Demo\Client>KernelConsole.exe \\.\pipe\kernel22.000008d4
Console Started 000007e8
setwritepipe

: _
```

## Demonstrator: Memory Leak Detection

```

C:\WINDOWS\system32\cmd.exe - KernelConsole.exe W:\pipe\kernel22.000008d4
setwritepipe
:show
Allocation ReturnAddr Allocator NumBytes Address ProcessId ThreadId TargetPid Module
0000000426 0x77d4eab6 HeapAlloc 0x00000100 0x000a9288 0x000008d4 0x00000224 0x000008d4 USER32.dll
0000000419 0x773de26d LocalAlloc 0x00000022 0x000af4a0 0x000008d4 0x00000224 0x000008d4 COMCTL32.dll
0000000418 0x01004908 LocalAlloc 0x00000002 0x009a0014 0x000008d4 0x00000224 0x000008d4 notepad.exe
0000000416 0x7742347c HeapReAlloc 0x00000600 0x000b9350 0x000008d4 0x00000224 0x000008d4 COMCTL32.dll
0000000415 0x773d61e2 LocalReAlloc 0x00000040 0x000b8540 0x000008d4 0x00000224 0x000008d4 COMCTL32.dll
0000000414 0x77c2c756 HeapReAlloc 0x000000e0 0x002c6d58 0x000008d4 0x00000224 0x000008d4 msvcrt.dll
0000000413 0x77c2c3c9 HeapAlloc 0x00000050 0x002c6470 0x000008d4 0x00000224 0x000008d4 msvcrt.dll
0000000412 0x77c2c3c9 HeapAlloc 0x00000048 0x002c6408 0x000008d4 0x00000224 0x000008d4 msvcrt.dll
0000000409 0x773de478 LocalAlloc 0x00000084 0x000a4f20 0x000008d4 0x00000224 0x000008d4 COMCTL32.dll
0000000407 0x77c2c3c9 HeapAlloc 0x000000dc 0x002c6c20 0x000008d4 0x00000224 0x000008d4 msvcrt.dll
0000000406 0x77c2c3c9 HeapAlloc 0x00000048 0x002c6bd0 0x000008d4 0x00000224 0x000008d4 msvcrt.dll
0000000404 0x77c2c3c9 HeapAlloc 0x00000080 0x002c6b28 0x000008d4 0x00000224 0x000008d4 msvcrt.dll
0000000402 0x7742aa68 LocalAlloc 0x00000008 0x000b22f8 0x000008d4 0x00000224 0x000008d4 COMCTL32.dll
0000000401 0x77c2c3c9 HeapAlloc 0x0000004c 0x002c66b0 0x000008d4 0x00000224 0x000008d4 msvcrt.dll
0000000398 0x77c2c3c9 HeapAlloc 0x00000018 0x002c6690 0x000008d4 0x00000224 0x000008d4 msvcrt.dll
0000000397 0x77c2c3c9 HeapAlloc 0x00000028 0x002c6640 0x000008d4 0x00000224 0x000008d4 msvcrt.dll
0000000395 0x77c2c3c9 HeapAlloc 0x00000048 0x002c65f0 0x000008d4 0x00000224 0x000008d4 msvcrt.dll
0000000393 0x774221f3 LocalAlloc 0x00000040 0x009a000c 0x000008d4 0x00000224 0x000008d4 COMCTL32.dll
0000000392 0x773f3fc9 HeapAlloc 0x00000128 0x000b7a10 0x000008d4 0x00000224 0x000008d4 COMCTL32.dll
0000000389 0x77c2c3c9 HeapAlloc 0x0000004c 0x002c5f68 0x000008d4 0x00000224 0x000008d4 msvcrt.dll
0000000388 0x77c2c3c9 HeapAlloc 0x000000dc 0x002c5e40 0x000008d4 0x00000224 0x000008d4 msvcrt.dll

```

## Demonstration: Memory Leak Detection

```

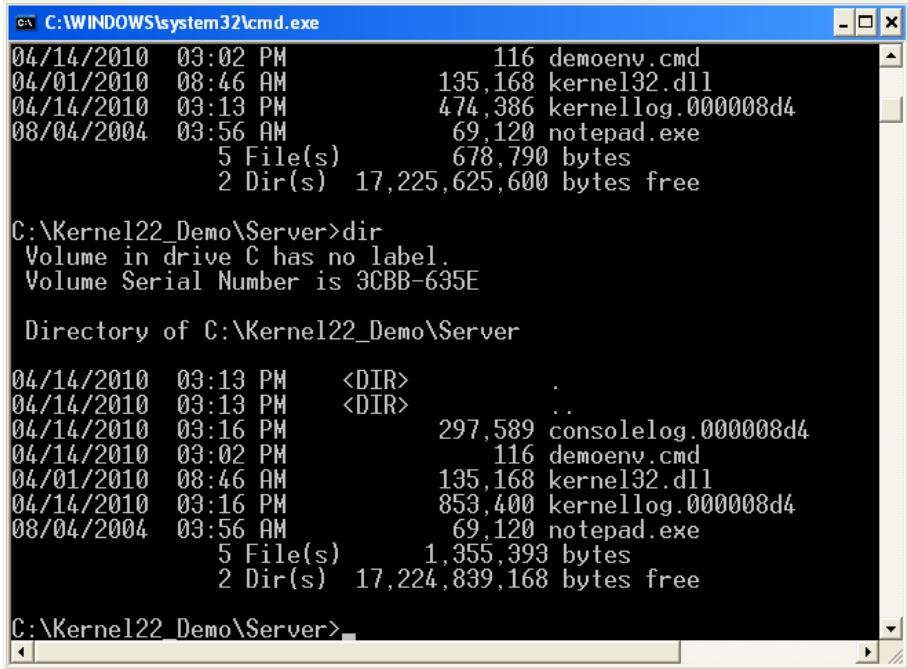
C:\WINDOWS\system32\cmd.exe - KernelConsole.exe \.\pipe\kernel22.000008d4
00000000 0x77c3a134 TlsAlloc 0x00000004 0x00000001 0x000008d4 0x00000224 0x000008d4 msvcrt.dll
show

:dump 426
Allocation ReturnAddr Allocator NumBytes Address ProcessId ThreadId TargetPid Module
-----
000000426 0x77d4eab6 HeapAlloc 0x00000100 0x000a9288 0x000008d4 0x00000224 0x000008d4 USER32.dll
000a9288: ee 01 2a 00 be 01 0e 00 01 00 00 00 78 00 30 00 :..*.....x.:.
000a9298: 30 00 30 00 30 00 30 00 38 00 64 00 34 00 20 00 :0.0.0.8.d.4.:.
000a92a8: 74 00 69 00 64 00 3a 00 20 00 30 00 78 00 30 00 :t.i.d.:.0x0.:.
000a92b8: 30 00 30 00 30 00 30 00 32 00 32 00 34 00 20 00 :0.0.0.2.2.4.:.
000a92c8: 6d 00 6f 00 64 00 75 00 6e 00 65 00 3a 00 20 00 :m.o.d.u.l.e.:.
000a92d8: 55 00 53 00 45 00 52 00 45 00 4e 00 56 00 2e 00 :U.S.E.R.E.M.U...
000a92e8: 64 00 6c 00 6c 00 5b 00 30 00 78 00 37 00 36 00 :d.l.l.f.0.x.7.6.:.
000a92f8: 39 00 63 00 35 00 32 00 31 00 31 00 5d 00 3a 00 :9.c.5.2.1.1.1.:.
000a9308: 20 00 6c 00 73 00 74 00 72 00 63 00 70 00 79 00 :.l.s.t.r.c.p.y.:.
000a9318: 57 00 3a 00 20 00 37 00 36 00 61 00 36 00 31 00 :W.:.7.6.a.6.1.:.
000a9328: 30 00 39 00 38 00 2c 00 20 00 4c 00 6f 00 63 00 :0.9.8...L.o.c.:.
000a9338: 61 00 6c 00 20 00 53 00 65 00 74 00 74 00 69 00 :a.l..S.e.t.t.i.:.
000a9348: 6e 00 67 00 73 00 5c 00 54 00 65 00 6d 00 70 00 :n.g.s.\.I.e.m.p.:.
000a9358: 6f 00 72 00 61 00 72 00 79 00 20 00 49 00 6e 00 :o.r.a.r.y..l.n.:.
000a9368: 74 00 65 00 72 00 6e 00 65 00 74 00 20 00 46 00 :t.e.r.n.e.t..F.:.
000a9378: 69 00 6c 00 65 00 73 00 0d 00 0a 00 00 00 00 00 :i.l.e.s.....:

dump
:
    
```



## Demonstration: Memory Leak Detection



```
C:\WINDOWS\system32\cmd.exe
04/14/2010 03:02 PM          116 demoenv.cmd
04/01/2010 08:46 AM    135,168 kernel32.dll
04/14/2010 03:13 PM    474,386 kernellog.000008d4
08/04/2004 03:56 AM     69,120 notepad.exe
          5 File(s)      678,790 bytes
          2 Dir(s)  17,225,625,600 bytes free

C:\Kernel22_Demo\Server>dir
Volume in drive C has no label.
Volume Serial Number is 3CBB-635E

Directory of C:\Kernel22_Demo\Server

04/14/2010 03:13 PM    <DIR>          .
04/14/2010 03:13 PM    <DIR>          ..
          297,589 consolelog.000008d4
04/14/2010 03:02 PM          116 demoenv.cmd
04/01/2010 08:46 AM    135,168 kernel32.dll
04/14/2010 03:16 PM    853,400 kernellog.000008d4
08/04/2004 03:56 AM     69,120 notepad.exe
          5 File(s)      1,355,393 bytes
          2 Dir(s)  17,224,839,168 bytes free

C:\Kernel22_Demo\Server>
```

**The End  
Thank You**

## **Kernel-22**

A Framework for Creating Analysis Tools

Mike McCarl

ICSALabs

[mmccarl@icsalabs.com](mailto:mmccarl@icsalabs.com)



Copyright 2009 Cybertrust. All Rights Reserved.