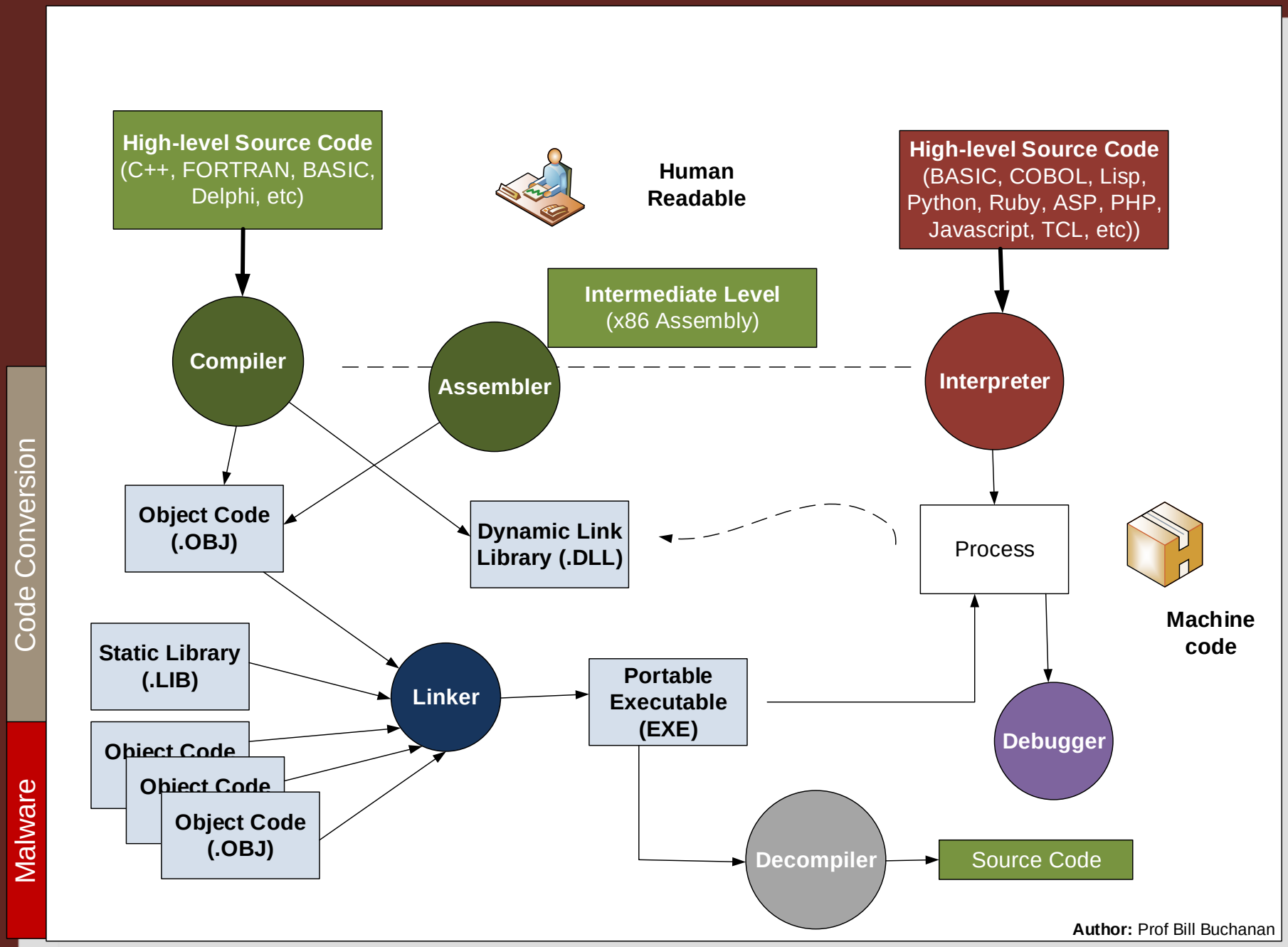
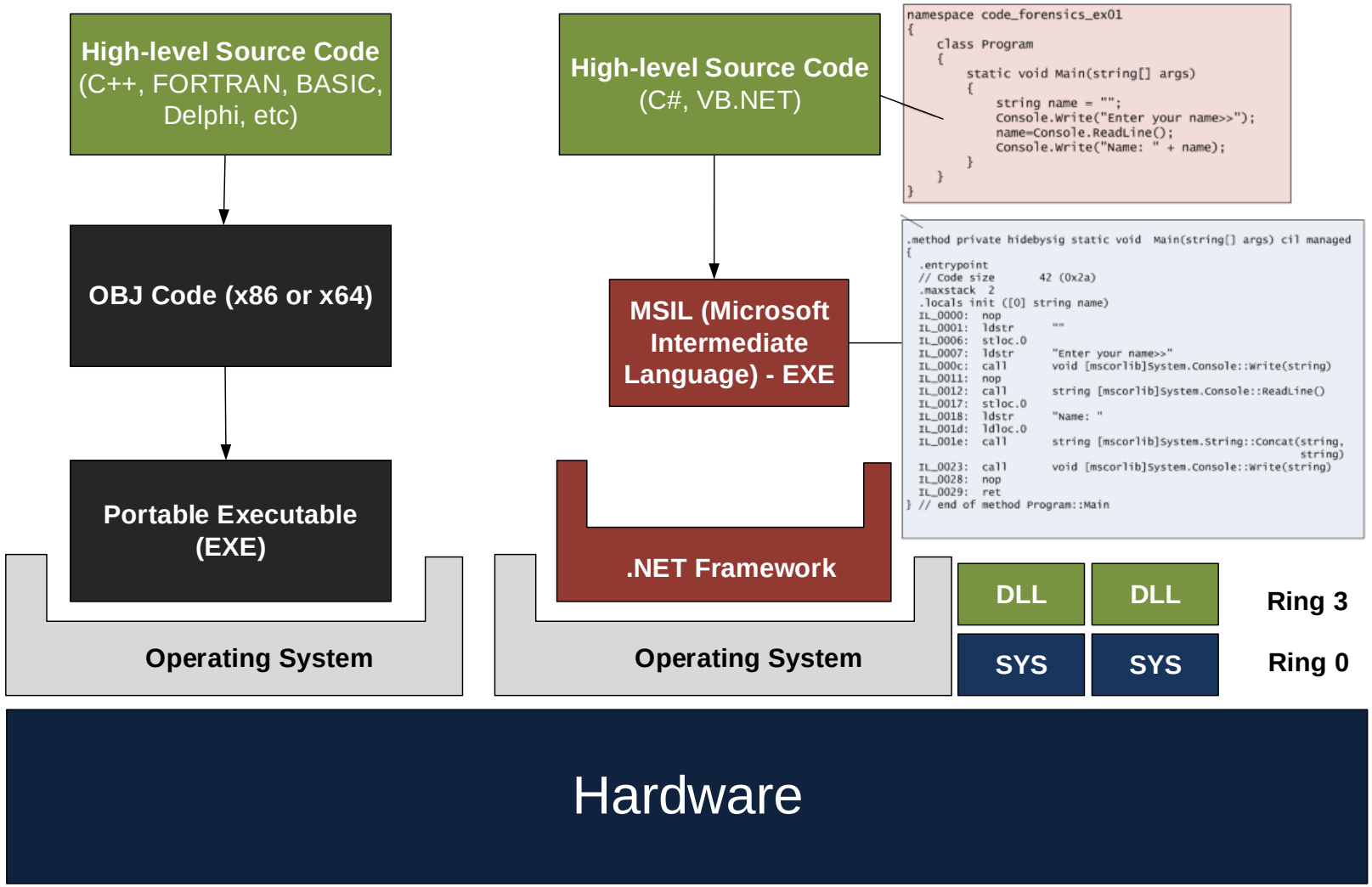
Malware Analysis

- Code Types.
- Taxonomy.
- Classifications.
- Malware Analysis (Static, Dynamic).
- Code Analysis.
- Forensic File Analysis.
- Real Malware Analysis.
- Exploit Kit.

Author: Prof Bill Buchanan

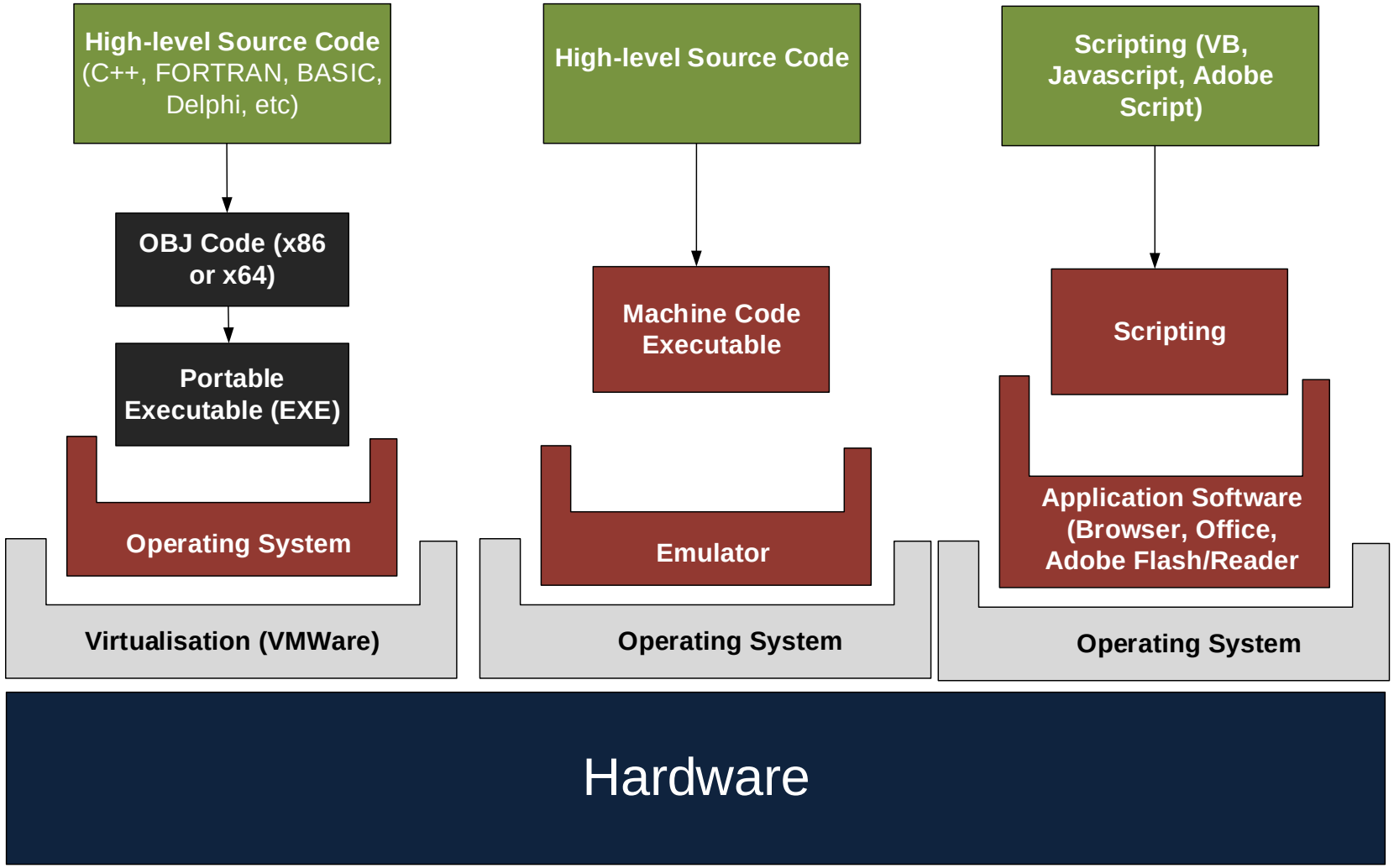

Code Conversion
Malware
High-level Source Code
(C++, FORTRAN, BASIC, Delphi, etc)
Human Readable
High-level Source Code
(BASIC, COBOL, Lisp, Python, Ruby, ASP, PHP, Javascript, TCL, etc))
Intermediate Level
(x86 Assembly)
Compiler
Assembler
Interpreter
Object Code
(.OBJ)
Dynamic Link Library (.DLL)
Process
Machine code
Static Library
(.LIB)
Linker
Portable Executable
(EXE)
Debugger
Object Code
Object Code
Object Code
(.OBJ)
Decompiler
Source Code
Author: Prof Bill Buchanan



Code Conversion

Malware

Author: Prof Bill Buchanan

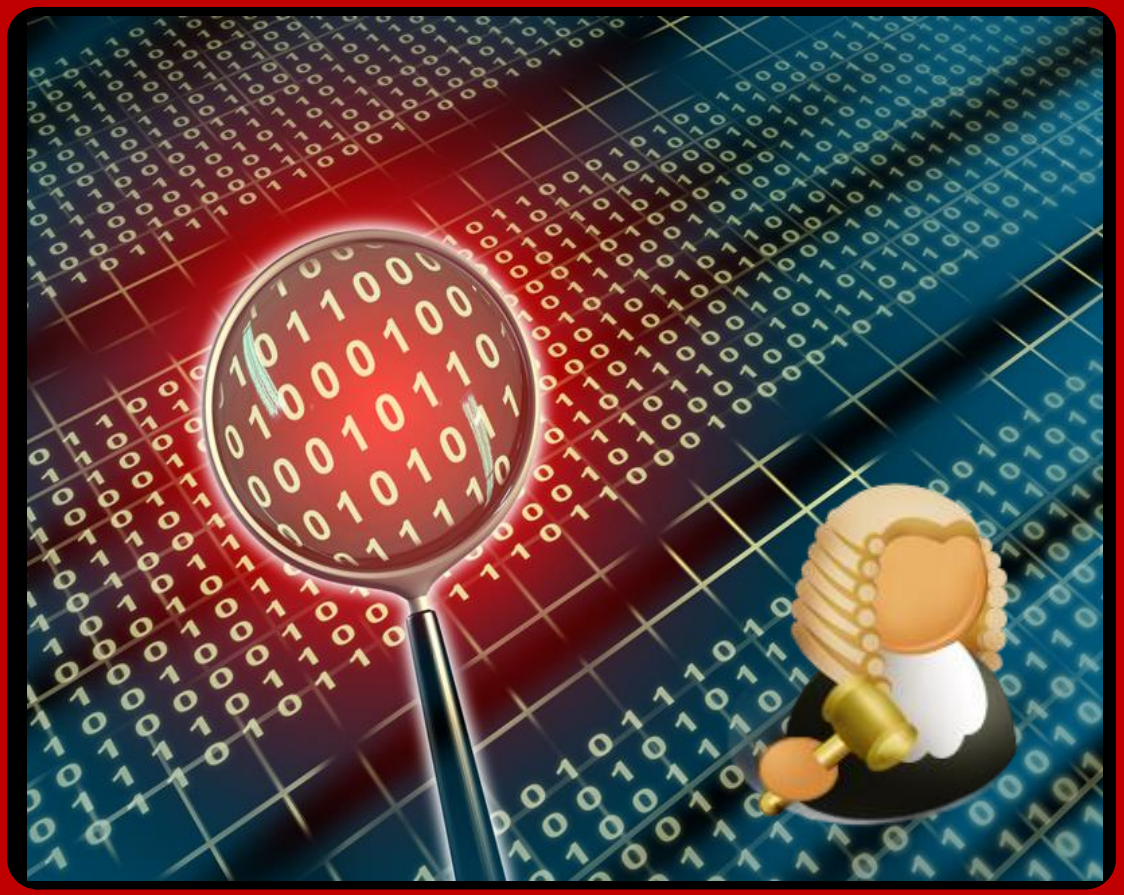


Code Conversion

Malware

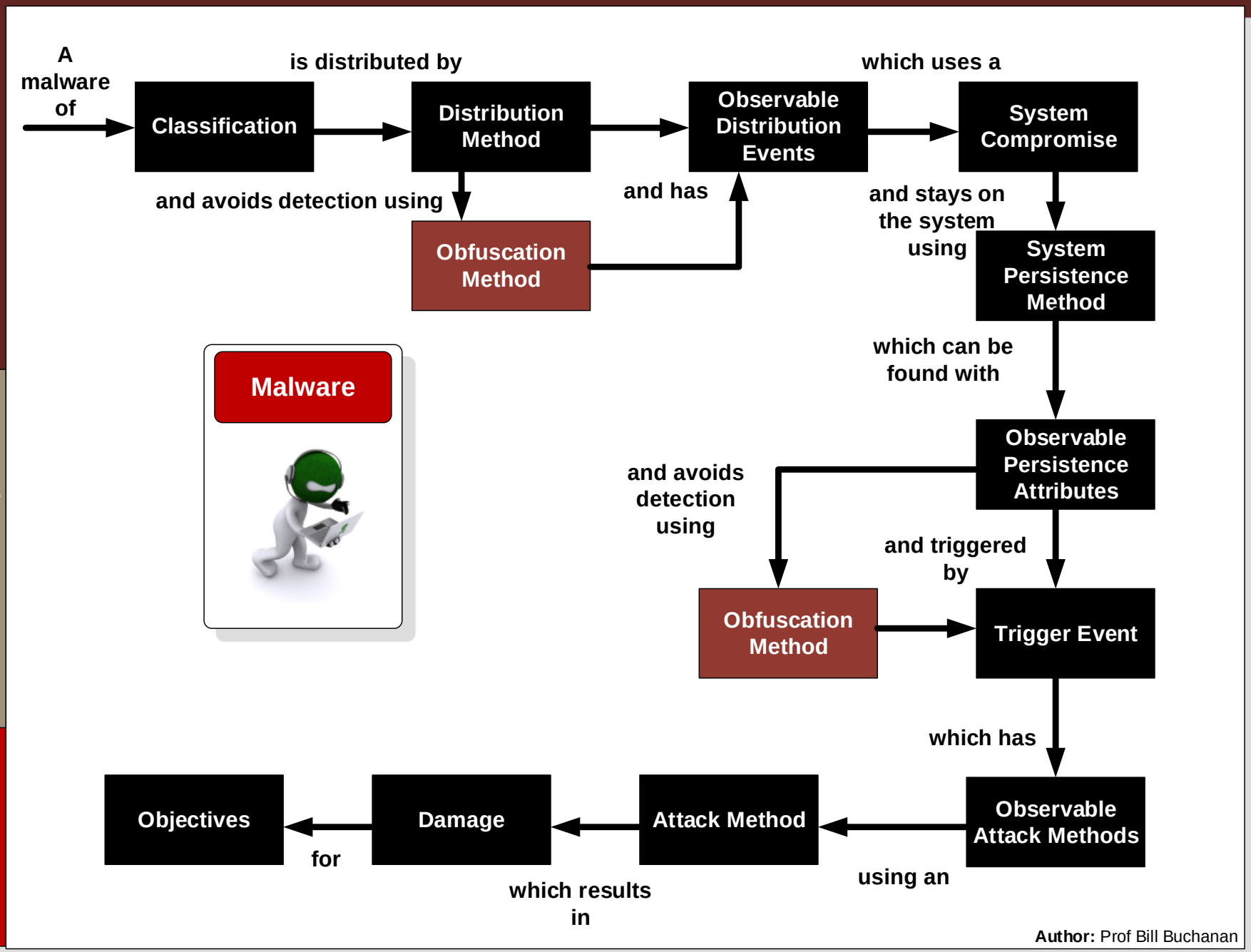
Author: Prof Bill Buchanan

Malware Analysis

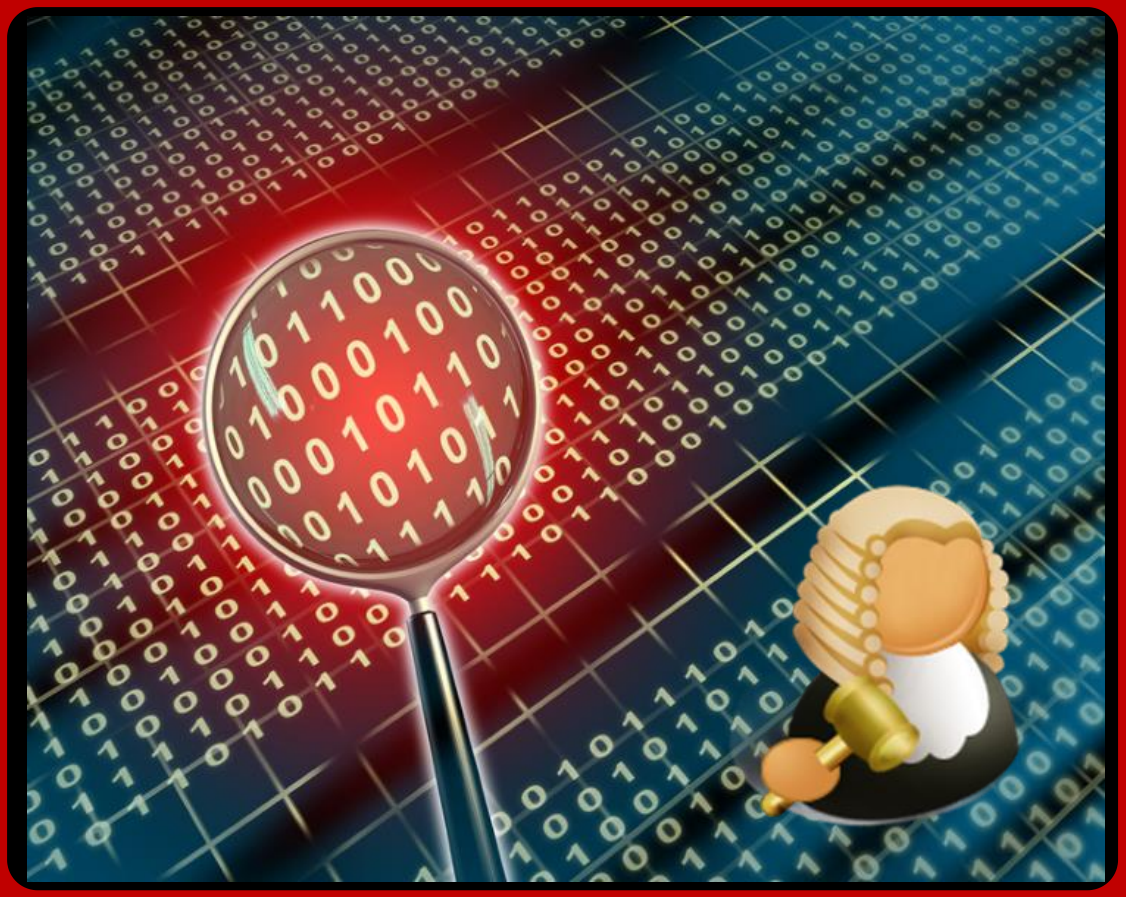


Taxonomy

Author: Prof Bill Buchanan


A malware of
Classification
is distributed by
Distribution Method
and avoids detection using
Obfuscation Method
and has
Observable Distribution Events
which uses a
System Compromise
and stays on the system using
System Persistence Method
which can be found with
Observable Persistence Attributes
and avoids detection using
Obfuscation Method
and triggered by
Trigger Event
which has
Observable Attack Methods
using an
Attack Method
which results in
Damage
for
Objectives
Malware
Taxonomy
Malware
Taxonomy
Author: Prof Bill Buchanan

Malware Analysis



Classifications

Author: Prof Bill Buchanan

A malware of → **Classification**

Backdoor

Backdoor. An intentional system hook left open to connect to.

Trojan horse

Trojan horses. This involves users running programs which look valid, but install an illicit program which will typically do damage to the host.

Ransomware

Ransomware. Ransomware involves extorting money from victims, such as for access a given site such as for pirate software.

Exploit

Exploit. This involves using compromising a vulnerability, such as Application (Adobe Reader), Operating System (Windows) or Browser (Mozilla).

Viruses

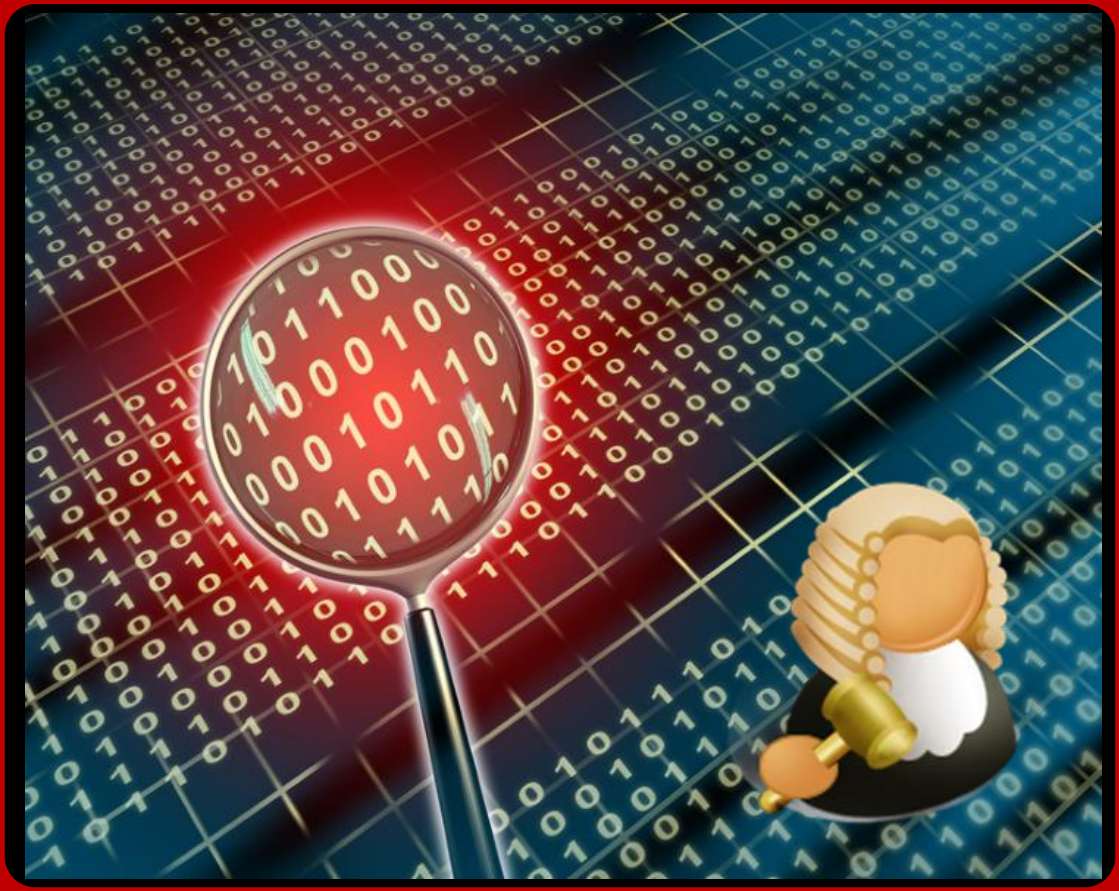
Viruses. This involves attaching program which self replicate themselves.

Worms

Malevolent worms. This involves a worm program which mutates in a given way which will eventually reduce the quality of service on the network, such as using up CPU resources or network bandwidth.

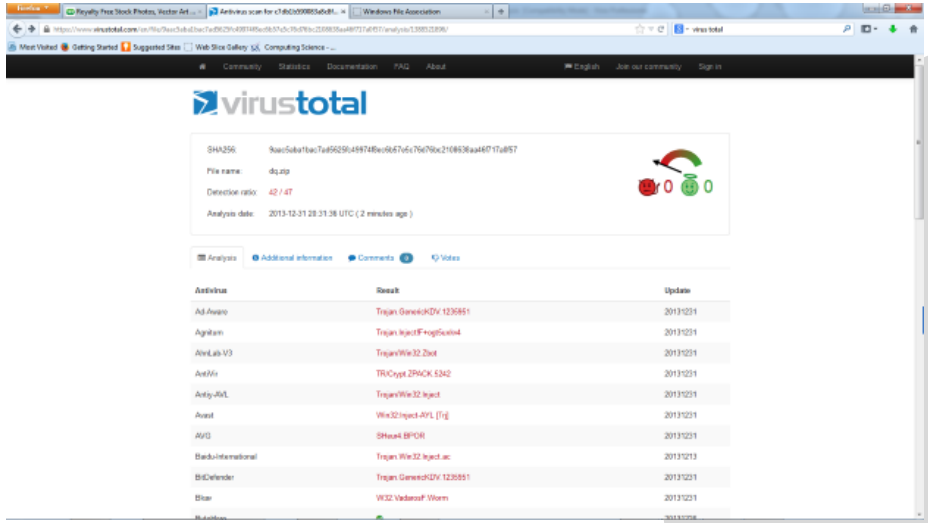
Author: Prof Bill Buchanan

Malware Analysis



Malware Analysis

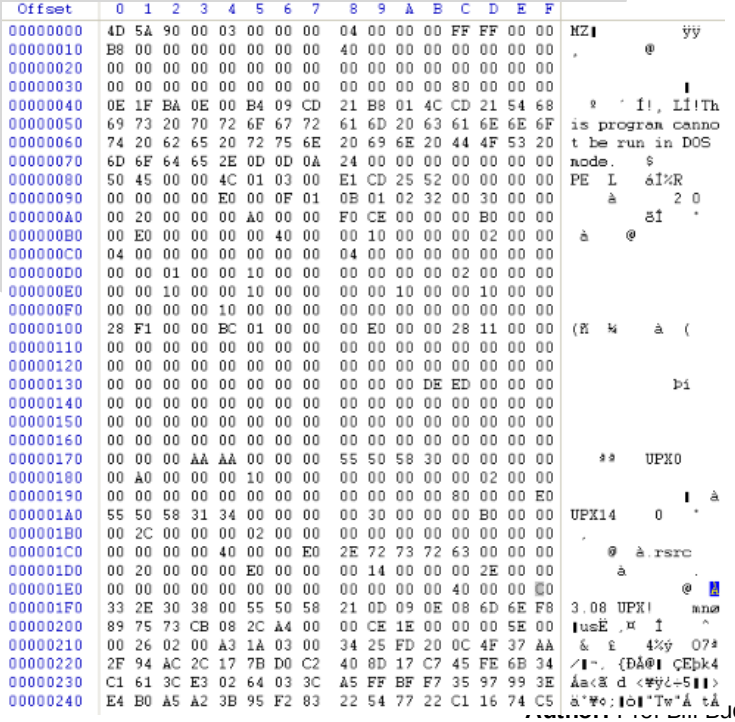
Author: Prof Bill Buchanan



Anti-virus signatures

Static Analysis

Strings from the malware

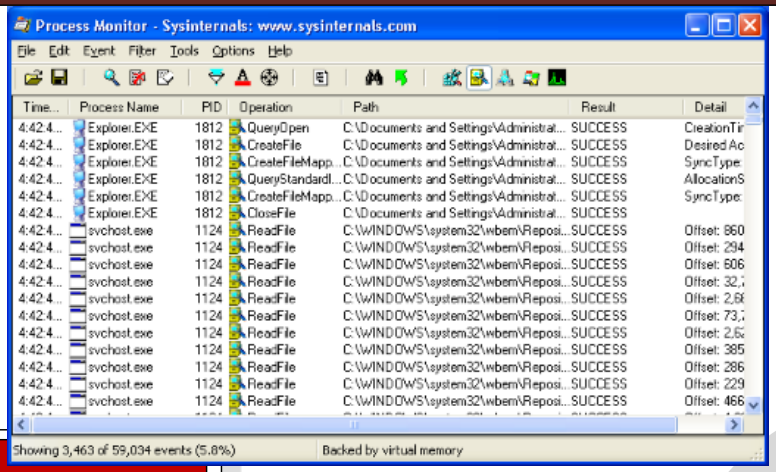


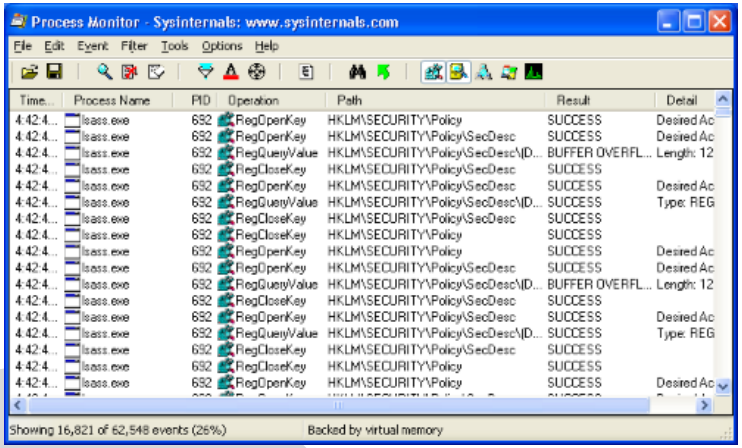
Analysis Tools

Malware

Analysis Tools

Malware





Malware

Dynamic Analysis

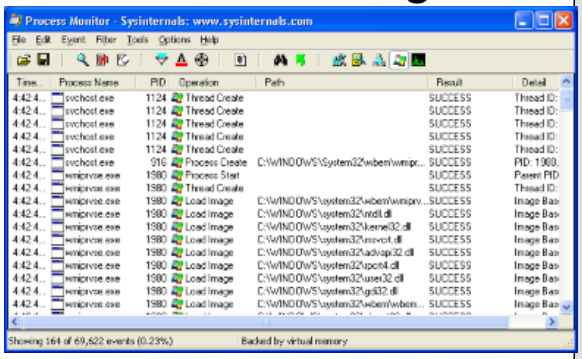
File Monitoring

Registry Monitoring

Network Monitoring

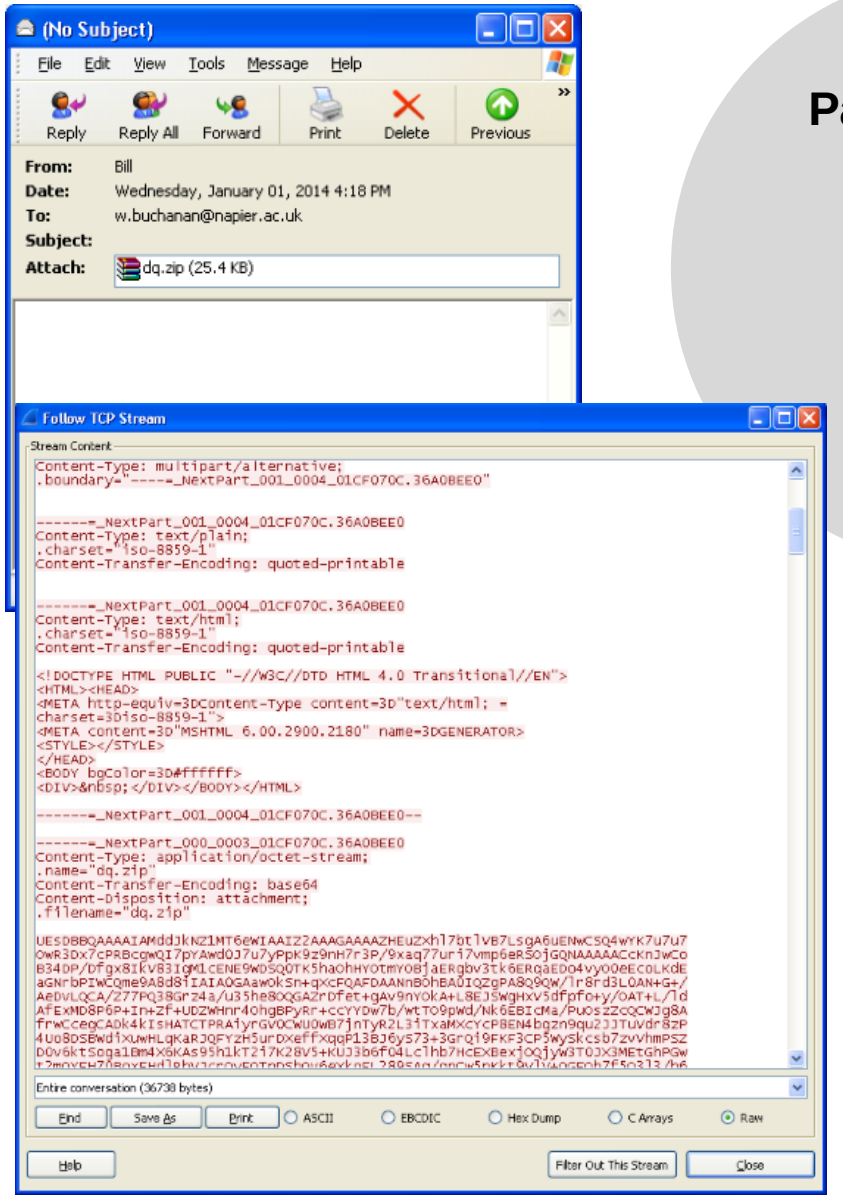
File Integrity check

Process Monitoring




```
C:\> fciv c:\ -r -type *.txt -sha1 -xml db.xml
C:\> fciv -list -sha1 -xml db.xml
C:\> fciv -v c:\ -sha1 -xml db.xml
Starting checksums verification : 01/01/2014 at 16h35'54
List of modified files:
-----------------------
c:\testing\1.txt
        Hash is        : f79cbaa6fb67d29a1636db52775de04d7479282e
        It should be   : 7c3f425725d7d6c48c5243c45d49b928b0bbf0e7

End Verification : 01/01/2014 at 16h35'54
```

Author: Prof Bill Buchanan



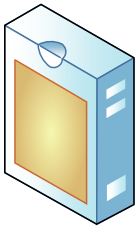
Payload Analysis

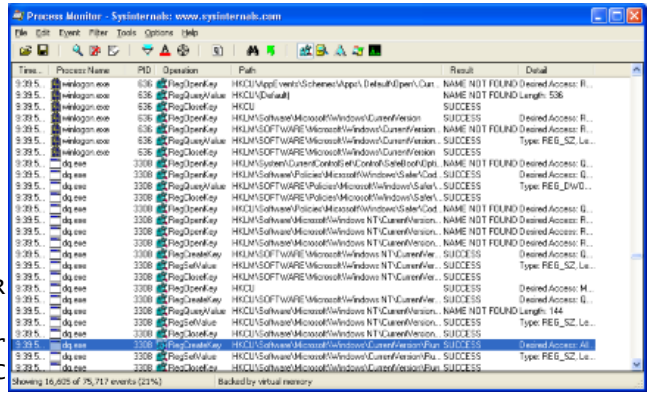
Dynamic Analysis

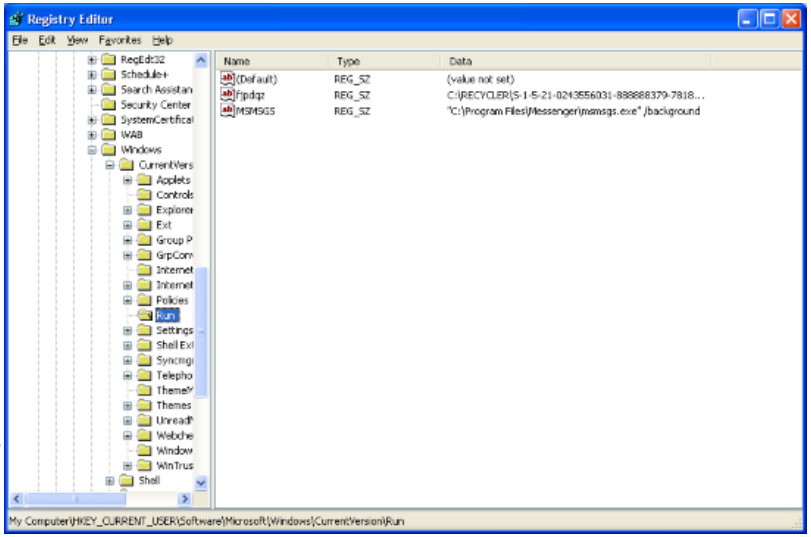
```
alert tcp any any -> any any
(content:"UEsDBBQAAAA";
msg:"Malware";sid:10000)
```

```
C:\Snort\bin\log>type alert.ids
[**] [1:10000:0] Malware [**]
[Priority: 0]
01/01-16:18:08.147386 192.168.47.146:1040
-> 192.168.47.134:25
TCP TTL:128 TOS:0x0 ID:167 IpLen:20
DgmLen:1500 DF
***A**** Seq: 0x95D103A3  Ack: 0x8108851
Win: 0xFEEF  TcpLen: 20
```

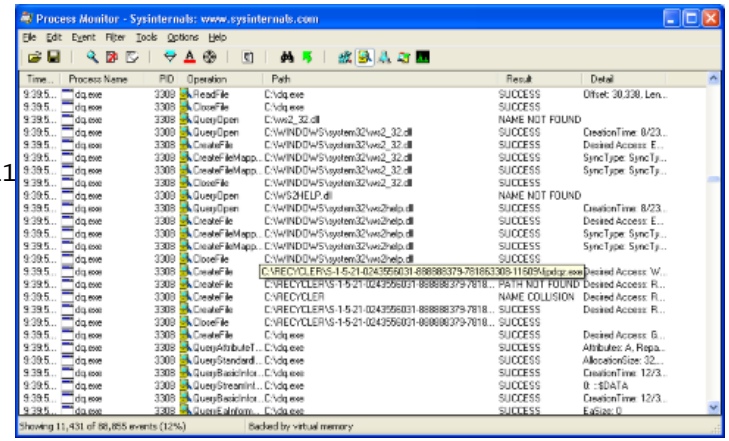
Author: Prof Bill Buchanan








```
C:\>cd RECYCLER

C:\RECYCLER>dir
 Directory of C

01/01/2014  09:03 PM <DIR>S-1-5-21-0243556031-888888379-781863
1609
               0 File(s)              0 bytes
               1 Dir(s)  18,497,097,728 bytes free

C:\RECYCLER>cd S-1-5-21-0243556031-888888379-781863308-11609

C:\RECYCLER\S-1-5-21-0243556031-888888379-781863308-11609>dir
 Volume in drive C has no label.
 Volume Serial Number is 606D-AF42

 Directory of C:\RECYCLER\S-1-5-21-0243556031-888888379-781863308-11

01/01/2014  09:03 PM    <DIR>          .
01/01/2014  09:03 PM    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)  18,497,097,728 bytes free

C:\RECYCLER\S-1-5-21-0243556031-888888379-781863308-11609>dir /ah
 Volume in drive C has no label.
 Volume Serial Number is 606D-AF42

 Directory of C:\RECYCLER\S-1-5-21-0243556031-888888379-781863308-11609

01/01/2014  09:03 PM                63 Desktop.ini
09/06/2013  10:46 AM            30,342 fjpdqz.exe
               2 File(s)         30,405 bytes
               0 Dir(s)  18,497,097,728 bytes free

C:\RECYCLER\S-1-5-21-0243556031-888888379-781863308-11609>
```

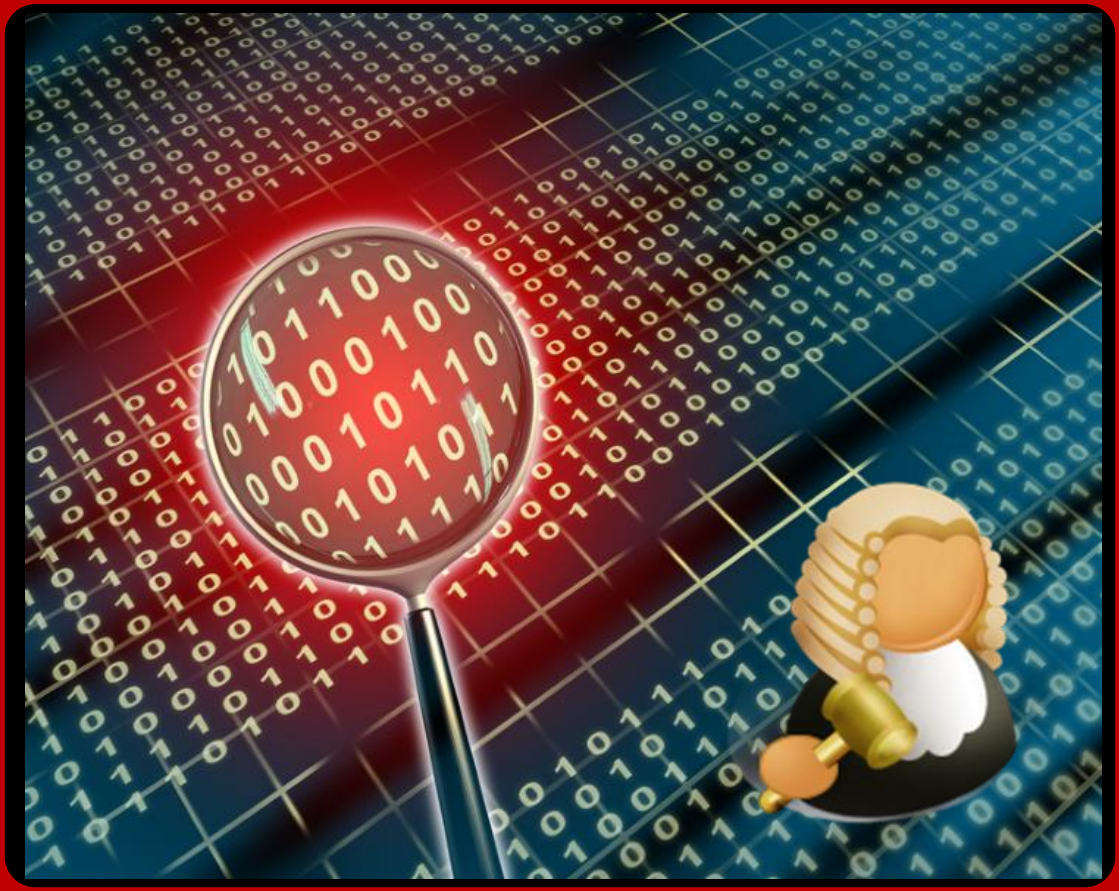


Analysis Tools

Malware

Author: Prof Bill Buchanan

Malware Analysis



Code Analysis

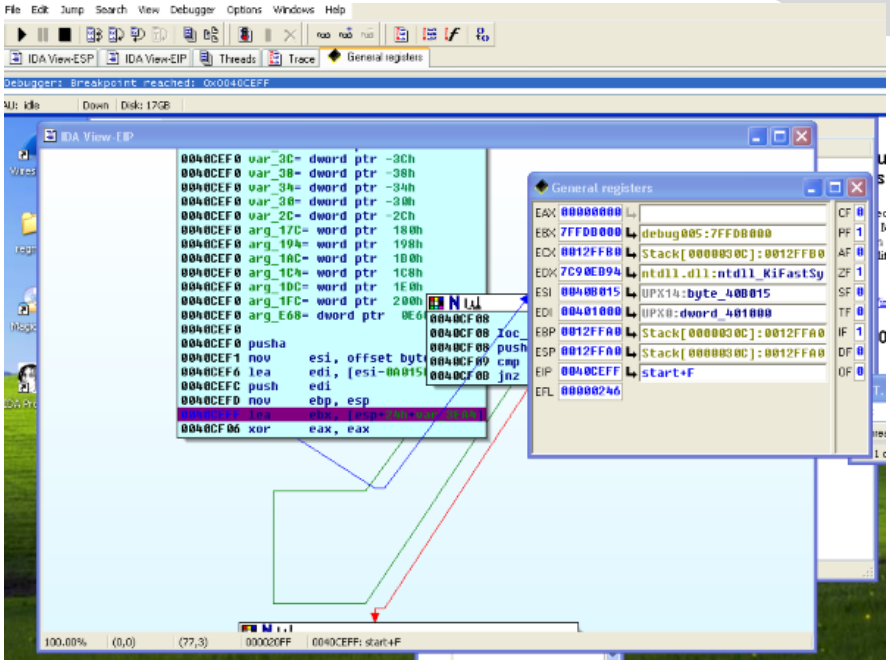
Author: Prof Bill Buchanan

Disassembly

```
.method private hidebysig static void  Main(string[] args) cil managed
{
  .entrypoint
  .custom instance void [mscorlib]System.STAThreadAttribute::.ctor() = ( 01 00 00 00 )
  // Code size       33 (0x21)
  .maxstack  2
  .locals init (string V_0)
  IL_0000:  ldstr     "What is your name\?"
  IL_0005:  call      void class [mscorlib]System.Console::Write(string)
  IL_000a:  call      string class [mscorlib]System.Console::ReadLine()
  IL_000f:  stloc.0
  IL_0010:  ldstr     "Hello "
  IL_0015:  ldloc.0
  IL_0016:  call      string string::Concat(string,
                                            string)
  IL_001b:  call      void class [mscorlib]System.Console::WriteLine(string)
  IL_0020:  ret
} // end of method a::Main
```

Code Analysis

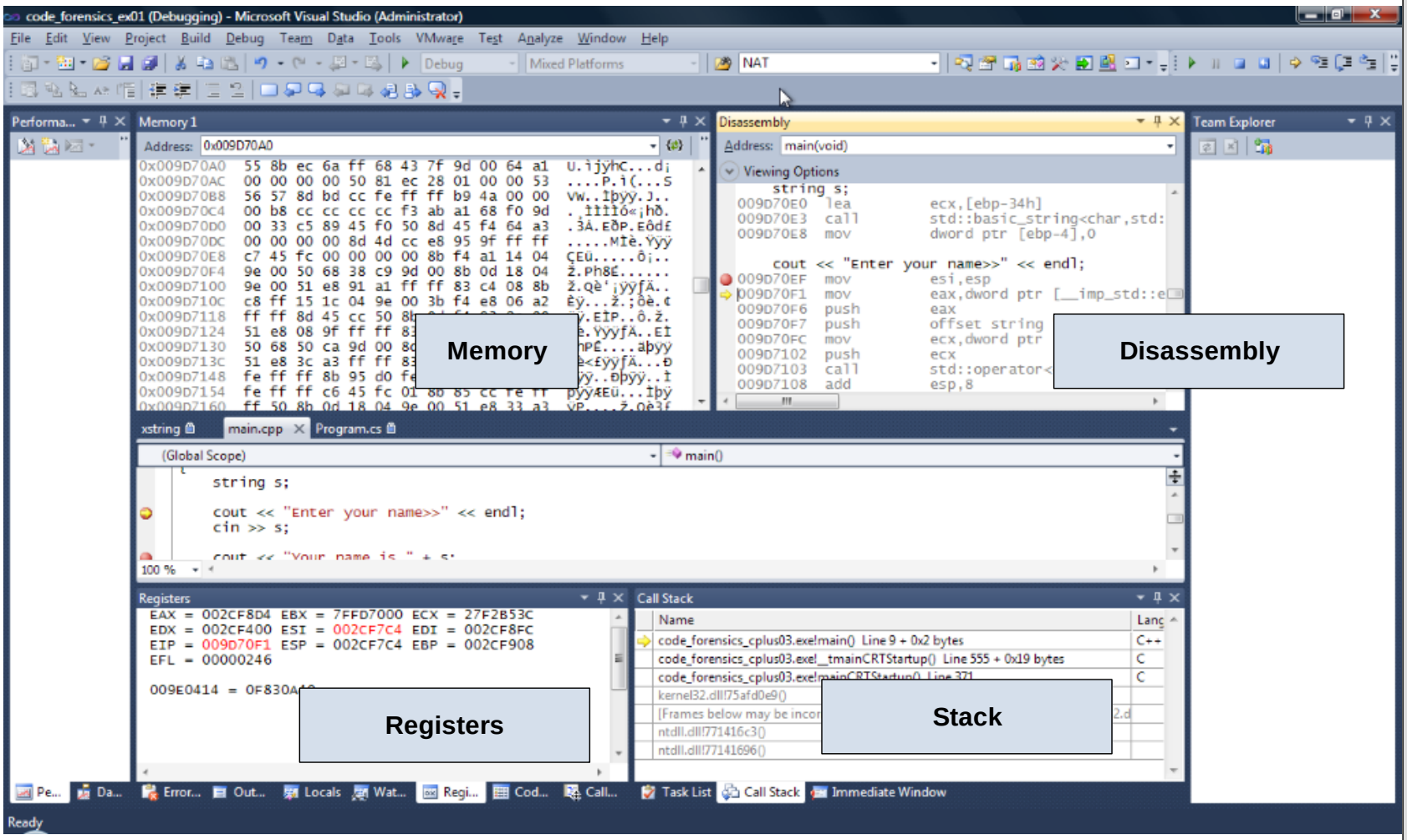
Debugging (IDA Pro)



Author: Prof Bill Buchanan

Analysis Tools

Malware



Author: Prof Bill Buchanan

```
int _tmain(int argc, _TCHAR* argv[])
{
00951380  push        ebp
00951381  mov         ebp,esp
00951383  sub         esp,0E4h
00951389  push        ebx
0095138A  push        esi
0095138B  push        edi
0095138C  lea         edi,[ebp-0E4h]
00951392  mov         ecx,39h
00951397  mov         eax,0CCCCCCCCh
0095139C  rep stos    dword ptr es:[edi]

int a=1;
0095139E  mov         dword ptr [a],1
int b=2;
009513A5  mov         dword ptr [b],2
int z;

z=a+b;
009513AC  mov         eax,dword ptr [a]
009513AF  add         eax,dword ptr [b]
009513B2  mov         dword ptr [z],eax
printf("%d",z);
009513B5  mov         esi,esp
009513B7  mov         eax,dword ptr [z]
009513BA  push        eax
009513BB  push        offset string "%d" (95573Ch)
009513C0  call        dword ptr [__imp__printf (9582B0h)]
009513C6  add         esp,8
009513C9  cmp         esi,esp
009513CB  call        @ILT+305(__RTC_CheckEsp) (951136h)
```

Mov	Move
Int	Interrupt
Add	Add values
Sub	Subtract values
Push	Put a value on the stack
Pop	Take it off the stack
Call	Jump to a function
Ret	Return to where called
Cmp	Compare two values
Je	Jump if equal to zero
Jne	Jump if not equal to zero
Jmp	Non-conditional jump

uP (x86 processor)

Author: Prof Bill Buchanan

```
int _tmain(int argc, _TCHAR* argv[])
{
00951380  push        ebp
00951381  mov         ebp,esp
00951383  sub         esp,0E4h
00951389  push        ebx
0095138A  push        esi
0095138B  push        edi
0095138C  lea         edi,[ebp-0E4h]
00951392  mov         ecx,39h
00951397  mov         eax,0CCCCCCCCh
0095139C  rep stos    dword ptr es:[edi]

int a=1;
0095139E  mov         dword ptr [a],1
int b=2;
009513A5  mov         dword ptr [b],2
int z;

z=a+b;
009513AC  mov         eax,dword ptr [a]
009513AF  add         eax,dword ptr [b]
009513B2  mov         dword ptr [z],eax
printf("%d",z);
009513B5  mov         esi,esp
009513B7  mov         eax,dword ptr [z]
009513BA  push        eax
009513BB  push        offset string "%d" (95573Ch)
009513C0  call        dword ptr [__imp__printf (9582B0h)]
009513C6  add         esp,8
009513C9  cmp         esi,esp
009513CB  call        @ILT+305(__RTC_CheckEsp) (951136h)
```

```
EAX = CCCCCCCC
EBX = 7FFDD000
ECX = 00000000
EDX = 00000001
ESI = 00000000
EDI = 0025F7F4
EIP = 0095139E
ESP = 0025F704
EBP = 0025F7F4
EFL = 00000202
```

```
EAX = CCCCCCCC
EBX = 7FFDD000
ECX = 00000000
EDX = 00000001
ESI = 00000000
EDI = 0025F7F4
EIP = 009513A5
ESP = 0025F704
EBP = 0025F7F4
EFL = 00000202
```

```
EAX = CCCCCCCC
EBX = 7FFDD000
ECX = 00000000
EDX = 00000001
ESI = 00000000
EDI = 0025F7F4
EIP = 009513AC
ESP = 0025F704
EBP = 0025F7F4
EFL = 00000202
```

```
EAX = 00000001
EBX = 7FFDD000
ECX = 00000000
EDX = 00000001
ESI = 00000000
EDI = 0025F7F4
EIP = 009513AF
ESP = 0025F704
EBP = 0025F7F4
EFL = 00000202
```

```
EAX = 00000003
EBX = 7FFDD000
ECX = 00000000
EDX = 00000001
ESI = 00000000
EDI = 0025F7F4
EIP = 009513B2
ESP = 0025F704
EBP = 0025F7F4
EFL = 00000206
```

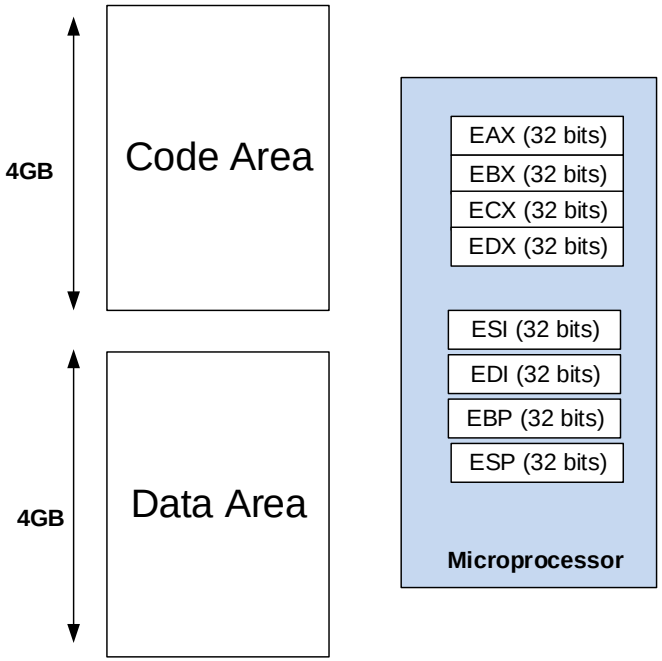
```
EAX = 00000003
EBX = 7FFDD000
ECX = 00000000
EDX = 00000001
ESI = 00000000
EDI = 0025F7F4
EIP = 009513B5
ESP = 0025F704
EBP = 0025F7F4
EFL = 00000206
```

uP (x86 processor)

Author: Prof Bill Buchanan

```
int _tmain(int argc, _TCHAR* argv[])
{
00951380  push        ebp
00951381  mov         ebp,esp
00951383  sub         esp,0E4h
00951389  push        ebx
0095138A  push        esi
0095138B  push        edi
0095138C  lea         edi,[ebp-0E4h]
00951392  mov         ecx,39h
00951397  mov         eax,0CCCCCCCCh
0095139C  rep stos    dword ptr es:[edi]

int a=1;
0095139E  mov         dword ptr [a],1
int b=2;
009513A5  mov         dword ptr [b],2
int z;

z=a+b;
009513AC  mov         eax,dword ptr [a]
009513AF  add         eax,dword ptr [b]
009513B2  mov         dword ptr [z],eax
printf("%d",z);
009513B5  mov         esi,esp
009513B7  mov         eax,dword ptr [z]
009513BA  push        eax
009513BB  push        offset string "%d" (95573Ch)
009513C0  call        dword ptr [__imp__printf (9582B0h)]
009513C6  add         esp,8
009513C9  cmp         esi,esp
009513CB  call        @ILT+305(__RTC_CheckEsp) (951136h)
```

uP (x86 processor)




```
EAX - Accumulator Register
EBX - Base Register
ECX - Counter Register
EDX - Data Register
ESI - Source Index
EDI - Destination Index
EBP - Base Pointer
ESP - Stack Pointer
EIP - Instruction Pointer
```

Author: Prof Bill Buchanan

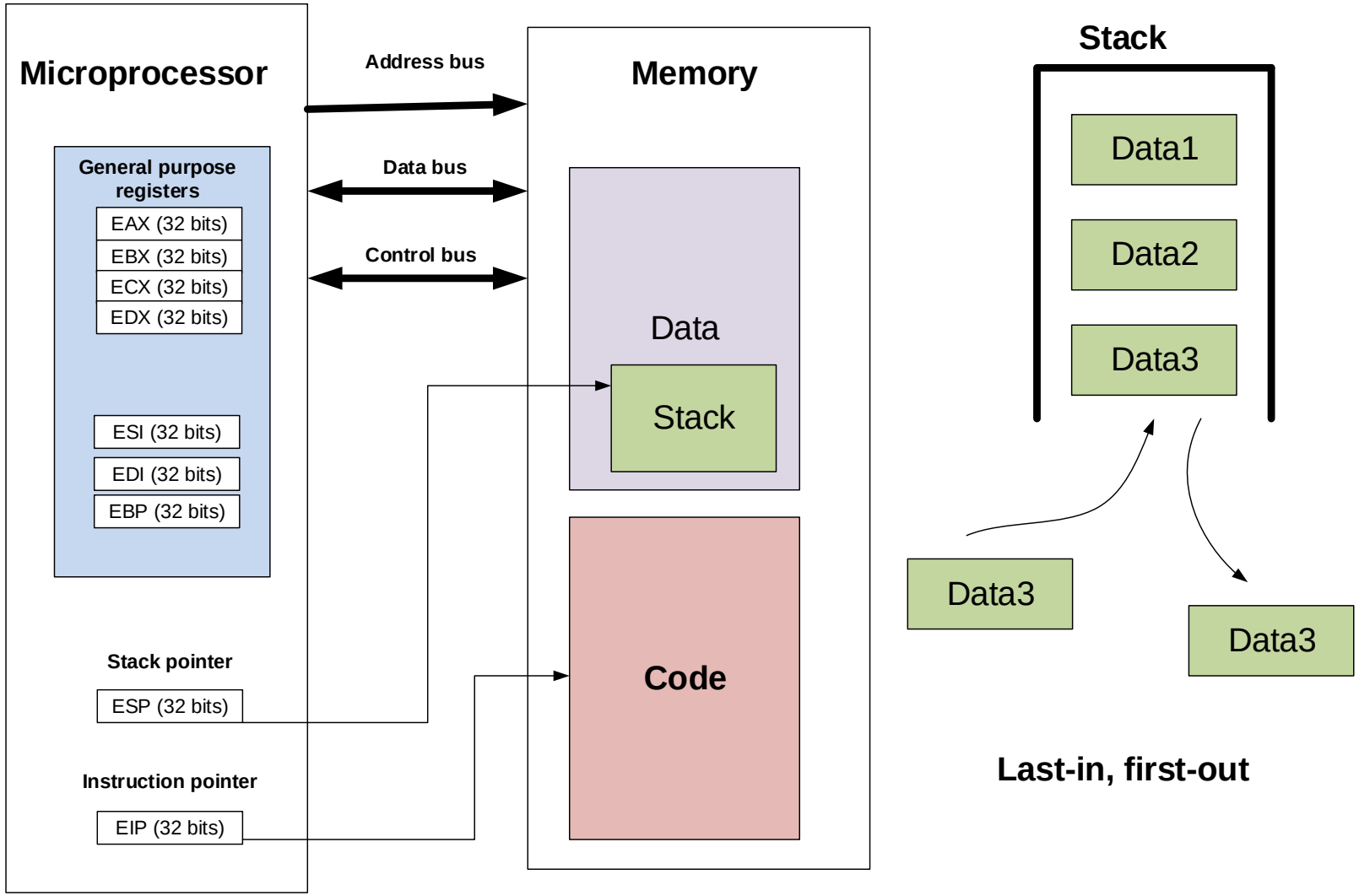
```
int _tmain(int argc, _TCHAR* argv[])
{
00BF13D0  push        ebp
00BF13D1  mov         ebp,esp
00BF13D3  sub         esp,0E4h
00BF13D9  push        ebx
00BF13DA  push        esi
00BF13DB  push        edi
00BF13DC  lea         edi,[ebp-0E4h]
00BF13E2  mov         ecx,39h
00BF13E7  mov         eax,0CCCCCCCCh
00BF13EC  rep stos    dword ptr es:[edi]

int a=1;
00BF13EE  mov         dword ptr [a],1
int b=2;
00BF13F5  mov         dword ptr [b],2
int z;

z=add(a,b);
00BF13FC  mov         eax,dword ptr [b]
00BF13FF  push        eax
00BF1400  mov         ecx,dword ptr [a]
00BF1403  push        ecx
00BF1404  call        add (0BF1091h)
00BF1409  add         esp,8
00BF140C  mov         dword ptr [z],eax
printf("%d",z);
00BF140F  mov         esi,esp
00BF1411  mov         eax,dword ptr [z]
00BF1414  push        eax
00BF1415  push        offset string "%d" (0BF573Ch)
00BF141A  call        dword ptr [__imp__printf (0BF82B0h)]
00BF1420  add         esp,8
00BF1423  cmp         esi,esp
00BF1425  call        @ILT+310(__RTC_CheckEsp) (0BF113Bh)


     return 0;
00BF142A  xor         eax,eax
}
```

```
int add(int c,int d)
{
    return (c+d);
}

int _tmain(int argc, _TCHAR* argv[])
{
int a=1;
int b=2;
int z;

z=add(a,b);
printf("%d",z);
    return 0;
}
```

Stack
(Last-in, first out)

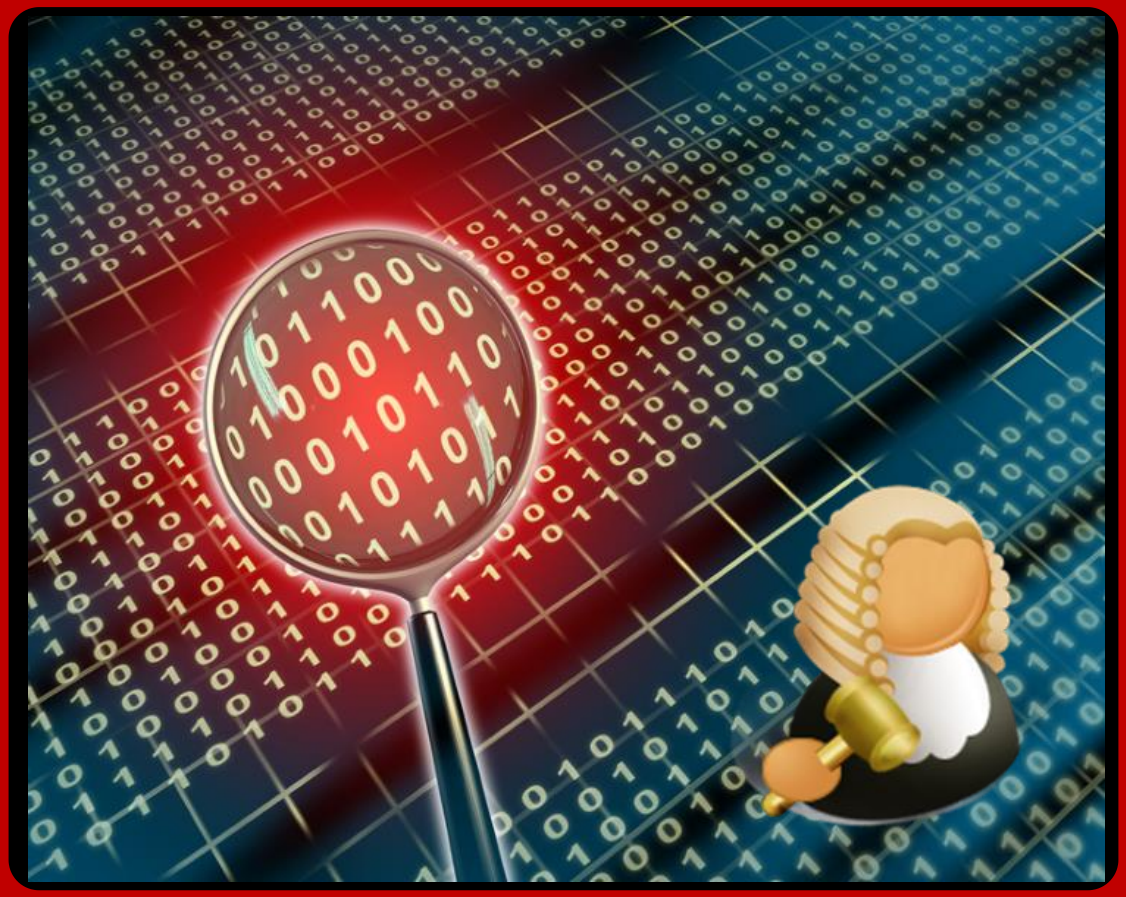
Push value onto the stack

```
int add(int c,int d)
{
000F1390  push        ebp
000F1391  mov         ebp,esp
000F1393  sub         esp,0C0h
000F1399  push        ebx
000F139A  push        esi
000F139B  push        edi
000F139C  lea         edi,[ebp-0C0h]
000F13A2  mov         ecx,30h
000F13A7  mov         eax,0CCCCCCCCh
000F13AC  rep stos    dword ptr es:[edi]
     return (c+d);
000F13AE  mov         eax,dword ptr [c]
000F13B1  add         eax,dword ptr [d]
}
000F13B4  pop         edi
000F13B5  pop         esi
000F13B6  pop         ebx
000F13B7  mov         esp,ebp
000F13B9  pop         ebp
000F13BA  ret
```



Author: Prof Bill Buchanan

Malware Analysis

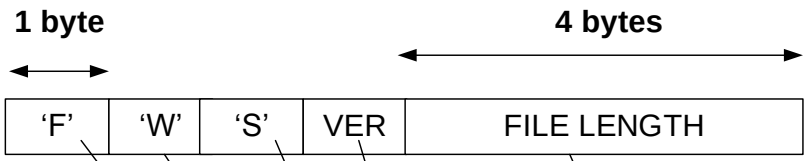


Forensic File Analysis

Author: Prof Bill Buchanan

Description	Extension	Magic Number
Adobe Illustrator	.ai	25 50 44 46 [%PDF]
Bitmap graphic	.bmp	42 4D [BM]
Class File	.class	CA FE BA BE
JPEG graphic file	.jpg	FFD8
JPEG 2000 graphic file	.jp2	0000000C6A5020200D0A [....jP..]
GIF graphic file	.gif	47 49 46 38 [GIF89]
TIF graphic file	.tif	49 49 [II]
PNG graphic file	.png	89 50 4E 47 .PNG
Photoshop Graphics	.psd	38 42 50 53 [8BPS]
Windows Meta File	.wmf	D7 CD C6 9A
MIDI file	.mid	4D 54 68 64 [MThd]
Icon file	.ico	00 00 01 00
MP3 file with ID3	.mp3	49 44 33 [ID3]
AVI video file	.avi	52 49 46 46 [RIFF]
Flash Shockwave	.swf	46 57 53 [FWS]
Flash Video	.flv	46 4C 56 [FLV]
Mpeg 4 video file	.mp4	00 00 00 18 66 74 79 70 6D 70 34 32 [....ftypmp42]
MOV video file	.mov	6D 6F 6F 76 [....moov]
Windows Video file	.wmv	30 26 B2 75 8E 66 CF
Windows Audio file	.wma	30 26 B2 75 8E 66 CF

Author: Prof Bill Buchanan




```
[00000000] 46 57 53 03 05 23 00 00   FWS..#..
[00000008] 58 00 7D 00 01 F4 00 00   X.}.....
[00000016] 0C 3C 00 43 02 FF FF FF   ...C....
[00000024] 7F 05 0C 20 00 00 01 00   ........
[00000032] FF D9 FF D8 FF D8 FF E0   ........
[00000040] 00 10 4A 46 49 46 00 01   ..JFIF..
```

Byte 0 [Should be 46 (F) or 43 (C)]: 46
Byte 1 [Should be 57 (W)]: 57
Byte 2 [Should be 53 (S)]: 53
Byte 3: Version: 03
Byte 4-7: File size (bytes): 8965
Next we read the next five bits to determine No of bit val: 11
No of bits used (for four values+5 bits): 49
No of bytes used for frame size: 7
Byte 15-16 Frame rate: 12
Byte 17-18 Frame count: 60
Byte 19-20 Tag ID: 9 Type: Control tag - Set Background colour
Data block size: 3 bytes

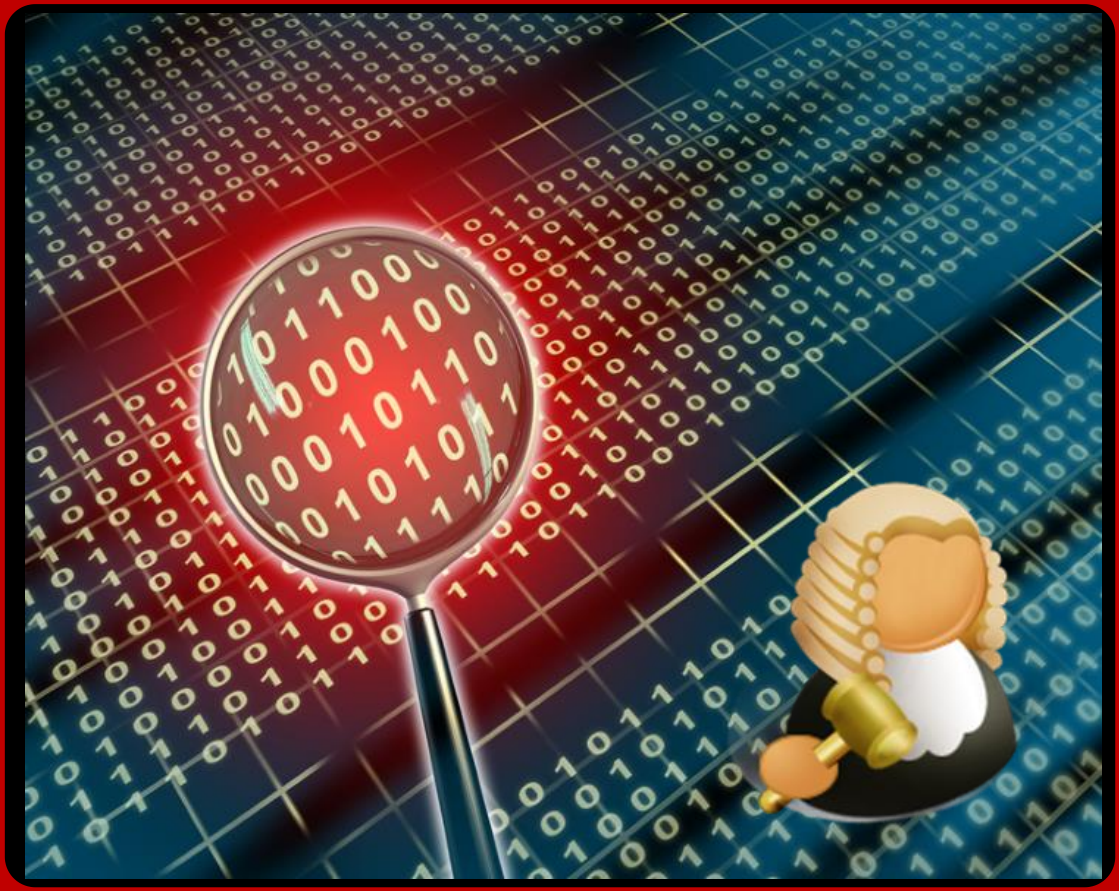
Data size

0000 0010 01 00 0011

10 bit SWF tag

Author: Prof Bill Buchanan

Malware Analysis



Exploit Kit

Author: Prof Bill Buchanan

HTML file contains Javascript which creates a document with Flash (tmp\flash.swf) which then loads malicious Javascript

```
function MDAC()
 {
   var p = document.createElement('object');
   p.setAttribute('id',p);
   p.setAttribute('classid','clsid:BD96C556-65A3-
11D0-983A-00C04FC29E36');
   try
   {
     var q = p.CreateObject('msxml2.XMLHTTP','');
     var r = p.CreateObject('Shell.Application','');
     var s = p.CreateObject('adodb.stream','');
..

     sv=new
ActiveXObject('ShockwaveFlash.ShockwaveFlas
h.9').GetVariable('$version');

  var vid = "<object width='300' height='300'
id='BridgeMovie'><param name='movie'
value='tmp/flash.swf'></param><param
name='allowScriptAccess'
value='sameDomain'></param><embed
src='tmp/flash.swf' name='BridgeMovie'
allowScriptAccess='sameDomain'
type='application/x-shockwave-flash' width='425'
height='355'></embed></object>";
```

CVE-2007-0071 - Integer overflow in Adobe Flash Player 9.0.115.0 and earlier, and 8.0.39.0 and earlier, allows remote attackers to execute arbitrary code via a crafted SWF file with a negative Scene Count value, which passes a signed comparison, is used as an offset of a NULL pointer, and triggers a buffer overflow – **LOADFLASH**.

CVE-2006-0003 - Unspecified vulnerability in the RDS.Dataspace ActiveX control, which is contained in ActiveX Data Objects (ADO) and distributed in Microsoft Data Access Components (MDAC) 2.7 and 2.8, allows remote attackers to execute arbitrary code via unknown attack vectors. **MDAC**.

CVE-2010-0806 Use-after-free vulnerability in the Peer Objects component (aka iepeers.dll) in Microsoft Internet Explorer 6, 6 SP1, and 7 allows remote attackers to execute arbitrary code via vectors involving access to an invalid pointer after the deletion of an object, as exploited in the wild in March 2010, aka "Uninitialized Memory Corruption Vulnerability." **MAKEHEAP**.

```
//ActionScript 3.0
//  package ie_fla
//    class MainTimeline
package ie_fla
{

    public dynamic class MainTimeline extends flash.display.MovieClip
    {
        public function MainTimeline()
        {
            super();
            addFrameScript(0, this.frame1);
            return;
        }

        function frame1():*
        {
            flash.external.ExternalInterface.call("FLASHSPRAY");
            flash.external.ExternalInterface.addCallback("sendFromJS",
this.recieveFromJS);
            flash.external.ExternalInterface.call("FLASHSPRAY");
            return;
        }

        public function Go():*
        {
 /// code missed out here
        }
        public function recieveFromJS(arg1:*):*
        {
            this.SWFstring = arg1;
            this.Go();
            return;
        }
        public var SWFstring:String;
    }
}
```

Analysis Tools

Malware

Malware Analysis

- Code Types.
- Taxonomy.
- Classifications.
- Malware Analysis (Static, Dynamic).
- Code Analysis.
- Forensic File Analysis.
- Real Malware Analysis.
- Exploit Kit.

Author: Prof Bill Buchanan