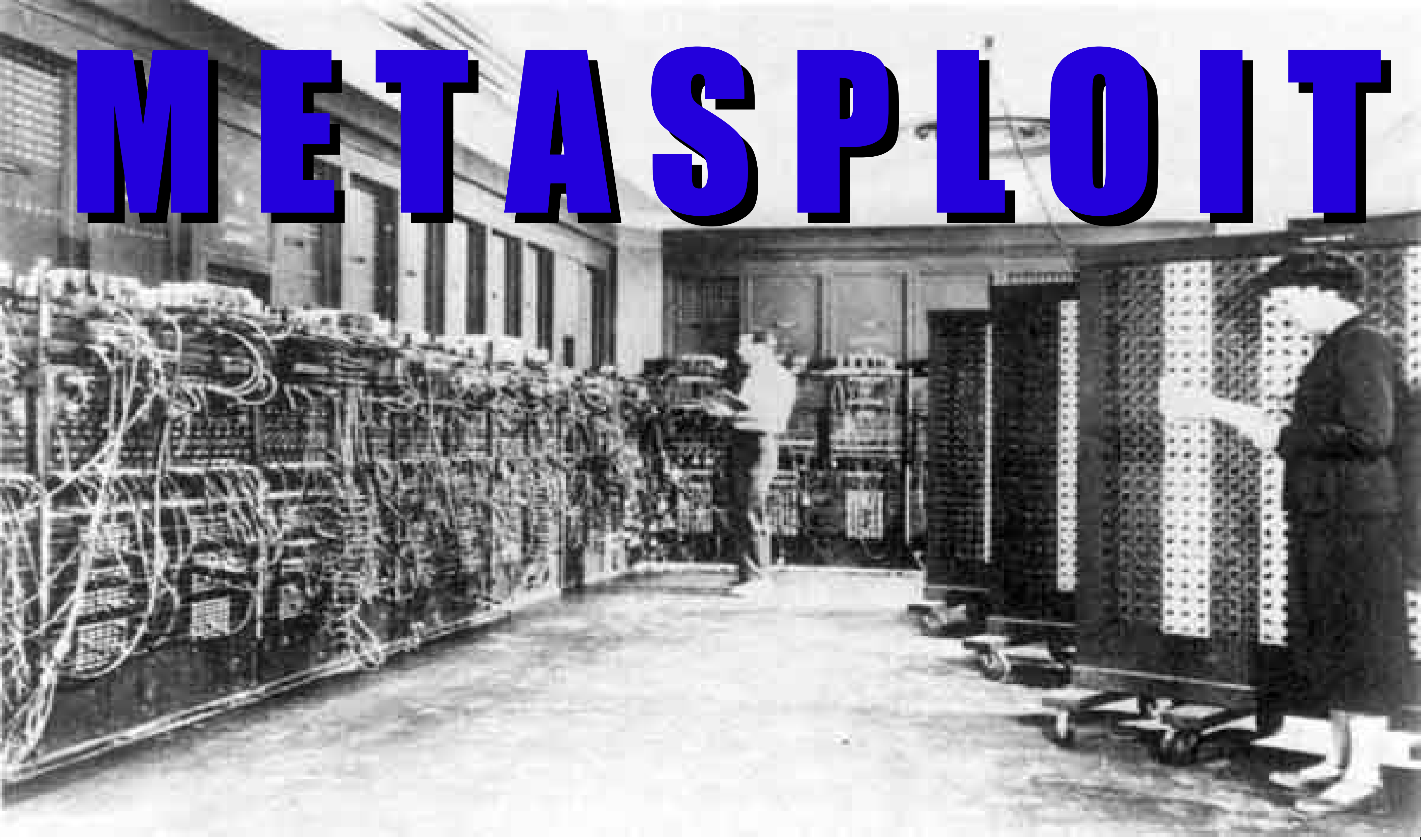


METASPLOIT

We have the technology

spoonm & hd moore – Redmond 2005

Who

- **Who are we?**
  - Lead developers of Metasploit
  - Vulnerability researchers
- **What do we do?**
  - HD is a cofounder of Digital Defense
  - Spoonm is a full-time student

- **What is this about?**
  - Exploit development process
  - Impact of Windows XP SP2
  - The Metasploit Framework
  - Technology demonstrations

Why

- **Why do we do this?**
  - Pen-testers need “clean” exploits
  - IDS vendors need a benchmark
  - Admins need to justify resources
  - Exploit research needs a kickstart
  - Fun :-)

Exploit Development
A Case Study

MS 2005

The Process

- **The exploit development process**
  - Disclosure
  - Analysis
  - Debugging
  - Development
  - Reliability

MS 2005

Case Study

- **Case Study: MS05-002**
  - Animated cursor buffer overflow
  - Affects mulitiple applications

Disclosure

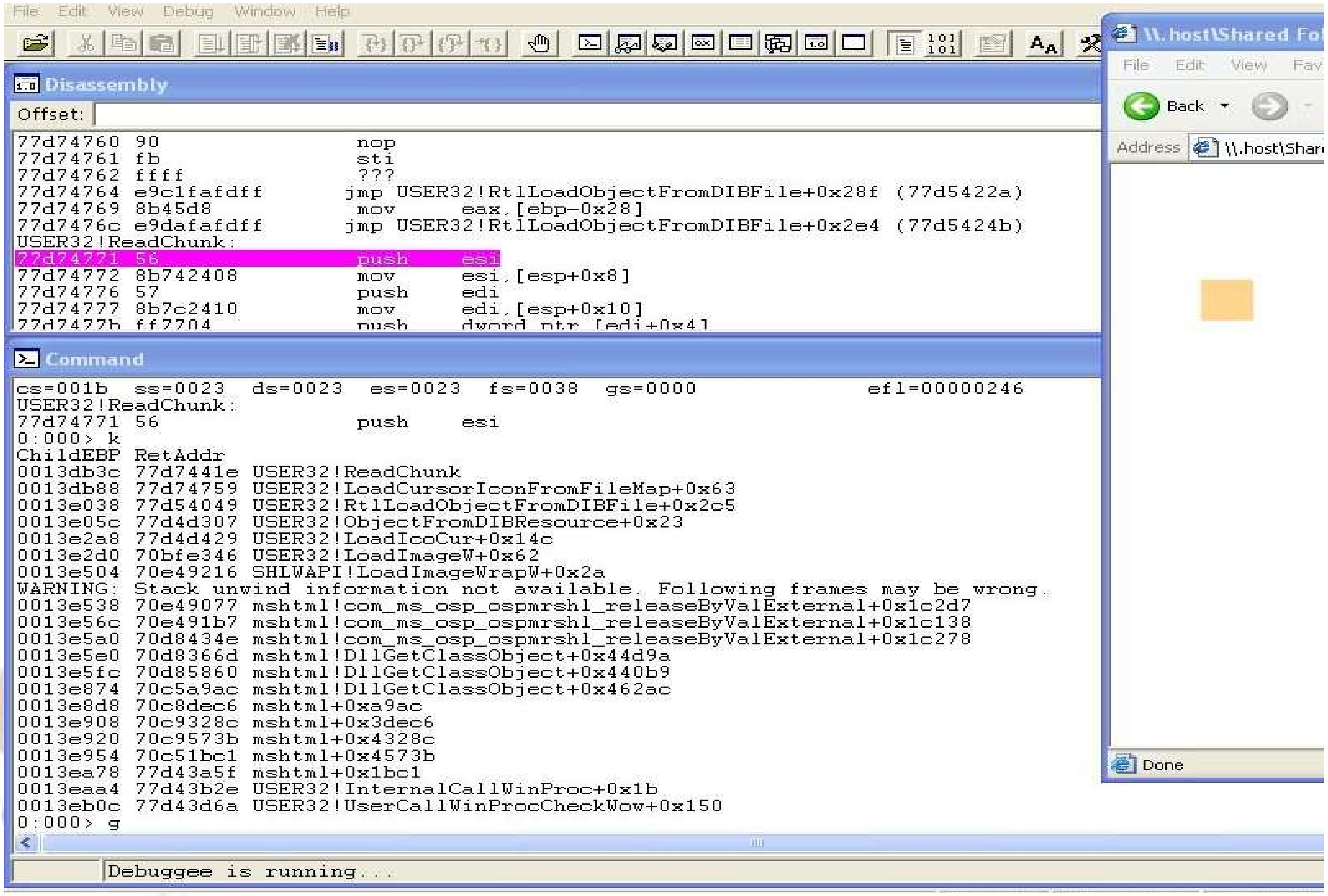
Microsoft discloses the bug
  - # MS05-002 contains no details
  - # Reversing patch is too involved
eYe's advisory
  - # Posted to security mailing lists
  - # Complete technical details :-)

Analysis

- Vulnerability is in user32.dll
  - Exploitable through multiple apps
  - Explorer, Outlook, IE main targets
- Multiple ways to exploit this
  - Web page in the Internet Zone
  - Directory listing in Explorer

Analysis

- Tracing the vulnerable code



```
77d74760 90              nop
77d74761 fb              sti
77d74762 ffff            ???
77d74764 e9c1fafdff     jmp USER32!RtlLoadObjectFromDIBFile+0x28f (77d5422a)
77d74769 8b45d8          mov     eax,[ebp-0x28]
77d7476c e9dafafdff     jmp USER32!RtlLoadObjectFromDIBFile+0x2e4 (77d5424b)
USER32!ReadChunk:
77d74771 56              push    esi
77d74772 8b742408        mov     esi,[esp+0x8]
77d74776 57              push    edi
77d74777 8b7c2410        mov     edi,[esp+0x10]
77d7477b ff7704          push    dword ptr [edi+0x4]
```

```
cs=001b  ss=0023  ds=0023  es=0023  fs=0038  gs=0000             efl=00000246
USER32!ReadChunk:
77d74771 56              push    esi
0:000> k
ChildEBP RetAddr
0013db3c 77d7441e USER32!ReadChunk
0013db88 77d74759 USER32!LoadCursorIconFromFileMap+0x63
0013e038 77d54049 USER32!RtlLoadObjectFromDIBFile+0x2c5
0013e05c 77d4d307 USER32!ObjectFromDIBResource+0x23
0013e2a8 77d4d429 USER32!LoadIcoCur+0x14c
0013e2d0 70bfe346 USER32!LoadImageW+0x62
0013e504 70e49216 SHLWAPI!LoadImageWrapW+0x2a
WARNING: Stack unwind information not available. Following frames may be wrong.
0013e538 70e49077 mshtml!com_ms_osp_ospmrshl_releaseByValExternal+0x1c2d7
0013e56c 70e491b7 mshtml!com_ms_osp_ospmrshl_releaseByValExternal+0x1c138
0013e5a0 70d8434e mshtml!com_ms_osp_ospmrshl_releaseByValExternal+0x1c278
0013e5e0 70d8366d mshtml!DllGetClassObject+0x44d9a
0013e5fc 70d85860 mshtml!DllGetClassObject+0x440b9
0013e874 70c5a9ac mshtml!DllGetClassObject+0x462ac
0013e8d8 70c8dec6 mshtml+0xa9ac
0013e908 70c9328c mshtml+0x3dec6
0013e920 70c9573b mshtml+0x4328c
0013e954 70c51bc1 mshtml+0x4573b
0013ea78 77d43a5f mshtml+0x1bc1
0013eaa4 77d43b2e USER32!InternalCallWinProc+0x1b
0013eb0c 77d43d6a USER32!UserCallWinProcCheckWow+0x150
0:000> g
```

Debuggee is running...

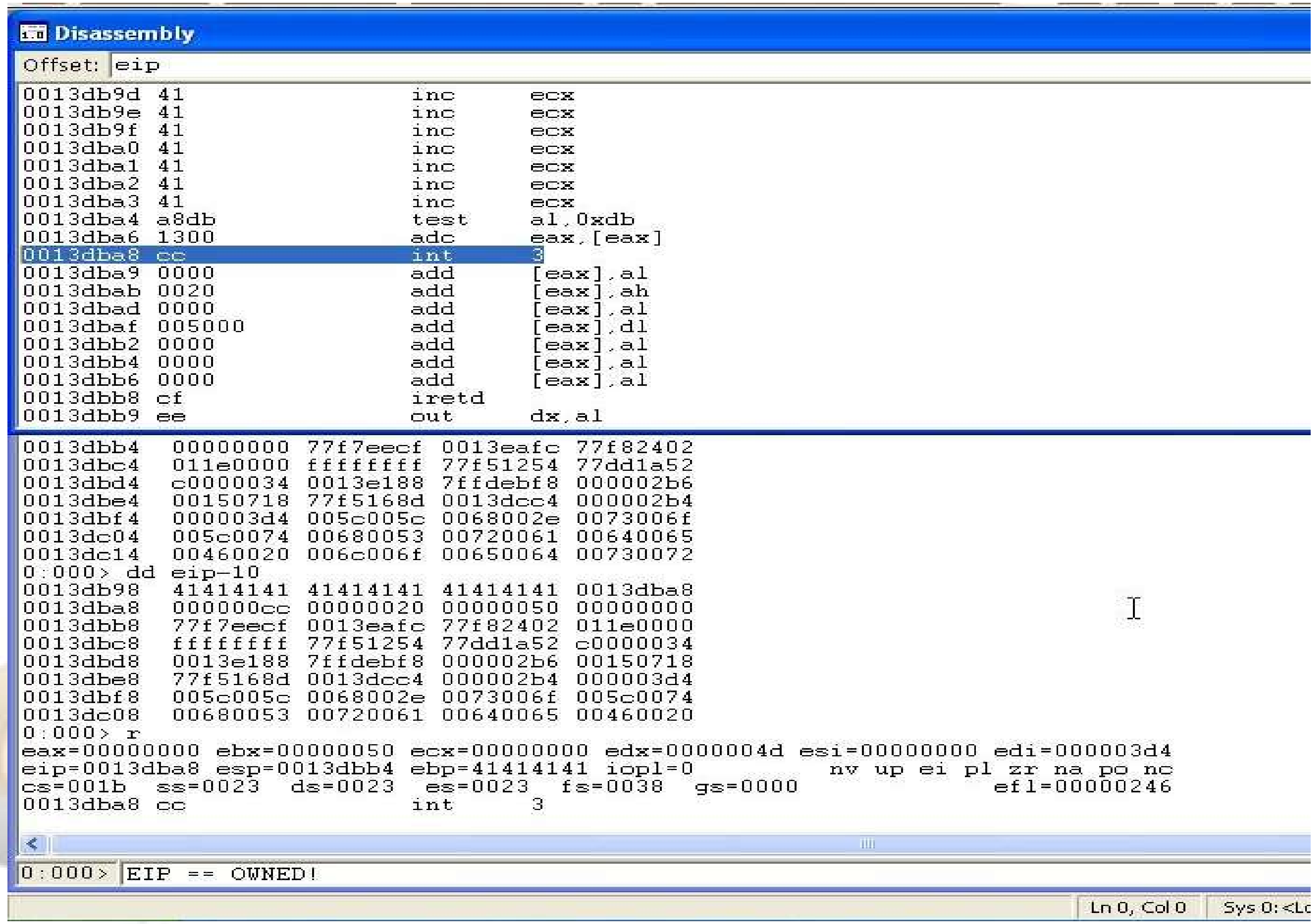
MS 2005

Debugging

- **Simple return address smash**
  - ESP register points back to data
  - Payload fits into the .ANI file
  - Return address should “jmp esp”
  - Set payload, set address, done.
  - Deliver via web page, email, UNC

Development

Code execution in WinDbg



```
0013db9d 41              inc     ecx
0013db9e 41              inc     ecx
0013db9f 41              inc     ecx
0013dba0 41              inc     ecx
0013dba1 41              inc     ecx
0013dba2 41              inc     ecx
0013dba3 41              inc     ecx
0013dba4 a8db            test    al,0xdb
0013dba6 1300            adc     eax,[eax]
0013dba8 cc              int     3
0013dba9 0000            add     [eax],al
0013dbab 0020            add     [eax],ah
0013dbad 0000            add     [eax],al
0013dbaf 005000          add     [eax],dl
0013dbb2 0000            add     [eax],al
0013dbb4 0000            add     [eax],al
0013dbb6 0000            add     [eax],al
0013dbb8 cf              iretd
0013dbb9 ee              out     dx,al
```

```
0013dbb4  00000000 77f7eecf 0013eafc 77f82402
0013dbc4  011e0000 ffffffff 77f51254 77dd1a52
0013dbd4  c0000034 0013e188 7ffdebf8 000002b6
0013dbe4  00150718 77f5168d 0013dcc4 000002b4
0013dbf4  000003d4 005c005c 0068002e 0073006f
0013dc04  005c0074 00680053 00720061 00640065
0013dc14  00460020 006c006f 00650064 00730072
0:000> dd eip-10
0013db98  41414141 41414141 41414141 0013dba8
0013dba8  000000cc 00000020 00000050 00000000
0013dbb8  77f7eecf 0013eafc 77f82402 011e0000
0013dbc8  ffffffff 77f51254 77dd1a52 c0000034
0013dbd8  0013e188 7ffdebf8 000002b6 00150718
0013dbe8  77f5168d 0013dcc4 000002b4 000003d4
0013dbf8  005c005c 0068002e 0073006f 005c0074
0013dc08  00680053 00720061 00640065 00460020
0:000> r
eax=00000000 ebx=00000050 ecx=00000000 edx=0000004d esi=00000000 edi=000003d4
eip=0013dba8 esp=0013dbb4 ebp=41414141 iopl=0         nv up ei pl zr na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=0038  gs=0000             efl=00000246
0013dba8 cc              int     3
```

```
0:000> EIP == OWNED!
```

Reliability

- **Return to ESP via ws2help.dll**
  - ws2help.dll is static across SPs
  - Address depends on the OS
  - Works fine with Internet Explorer
  - Doesn't always work with Explorer
  - Can fingerprint via User-Agent
  - Address found by Opcode DB

Service Pack 2

The Run Down

- A step in the right direction...
  - Too early to judge effectiveness
  - Third-party apps unaffected
  - SEH overwrites still possible
  - Heap protection weaknesses
  - DEP is mostly irrelevant

Third Parties

- **Third-party applications**
  - Not upgrading to new VS
  - Everyone runs 3rd party software
  - SP2 mechanisms do very little
- **Application Specific**
  - App specific exploit vector
  - Each bug leads to EIP differently

SEH Structures

- **Exception record on stack**

```
typedef struct _EXCEPTION_REGISTRATION
{
        struct _EXCEPTION_REGISTRATION* prev;
        PEXCEPTION_HANDLER              handler;
} EXCEPTION_REGISTRATION, *PEXCEPTION_REGISTRATION;
```

- **Exception handler**

```
EXCEPTION_DISPOSITION
__cdecl _except_handler(
    struct _EXCEPTION_RECORD *ExceptionRecord,
    void * EstablisherFrame,
    struct _CONTEXT *ContextRecord,
    void * DispatcherContext
);
```

Third Parties

- SEH frame overwrites
  - Return to 3rd party images (.exe)
  - pop/pop/ret is plentiful
  - Can't return to MS .exe or .dll
- Return address overwrites
  - Can still return to MS mappings
  - Returning to code not as nice as SEH

Summary

- **Summary**
  - A huge boost for the home user
  - Microsoft apps benefit the most
  - Third-party software is wide open
  - Limited impact on exploit writers

```
                __.                       .__.        .__. __.
  _____   _____/  |______    ____________ |  |   ____ |__|/  |_
 /     \_/ __ \   __\__  \  /  ___/\____ \|  |  /  _ \|  \   __\
|  Y Y  \  ___/|  |  / __ \_\___ \ |  |_> >  |_(  <_> )  ||  |
|__|_|  /\___  >__| (____  /____  >|   __/|____/\____/|__||__|
      \/     \/  v2.2    \/     \/ |__|
```

Metasploit Framework

MS 2005

Introduction

- **The Metasploit Framework**
  - Open source exploit framework
  - Exploit development platform
  - Written in Perl scripting language
  - Runs on most modern platforms
  - Designed for exploit research

Exploits

- **Exploits, exploits, exploits!**
  - Win32, MacOS, Linux, Solaris
  - DCOM, LSASS, MSSQL, Apache
  - Arkeia, BrightStor, Veritas, IIS
  - Samba, Squid, Unreal Tournament
- **Heavily tested, mostly reliable :-)**
- **Public version has ~60 exploits**

Payloads

- **Tiny chunks of assembly code**
  - Between 30 and 400 bytes long
  - Shells: bind, reverse, findsock
  - DLL injection, user-land execve
- **Multiple architectures and OSs**
  - IA32 (x86), SPARC, PPC, MIPS
  - Win32, Linux, Solaris, IRIX, MacOS

MS 2005

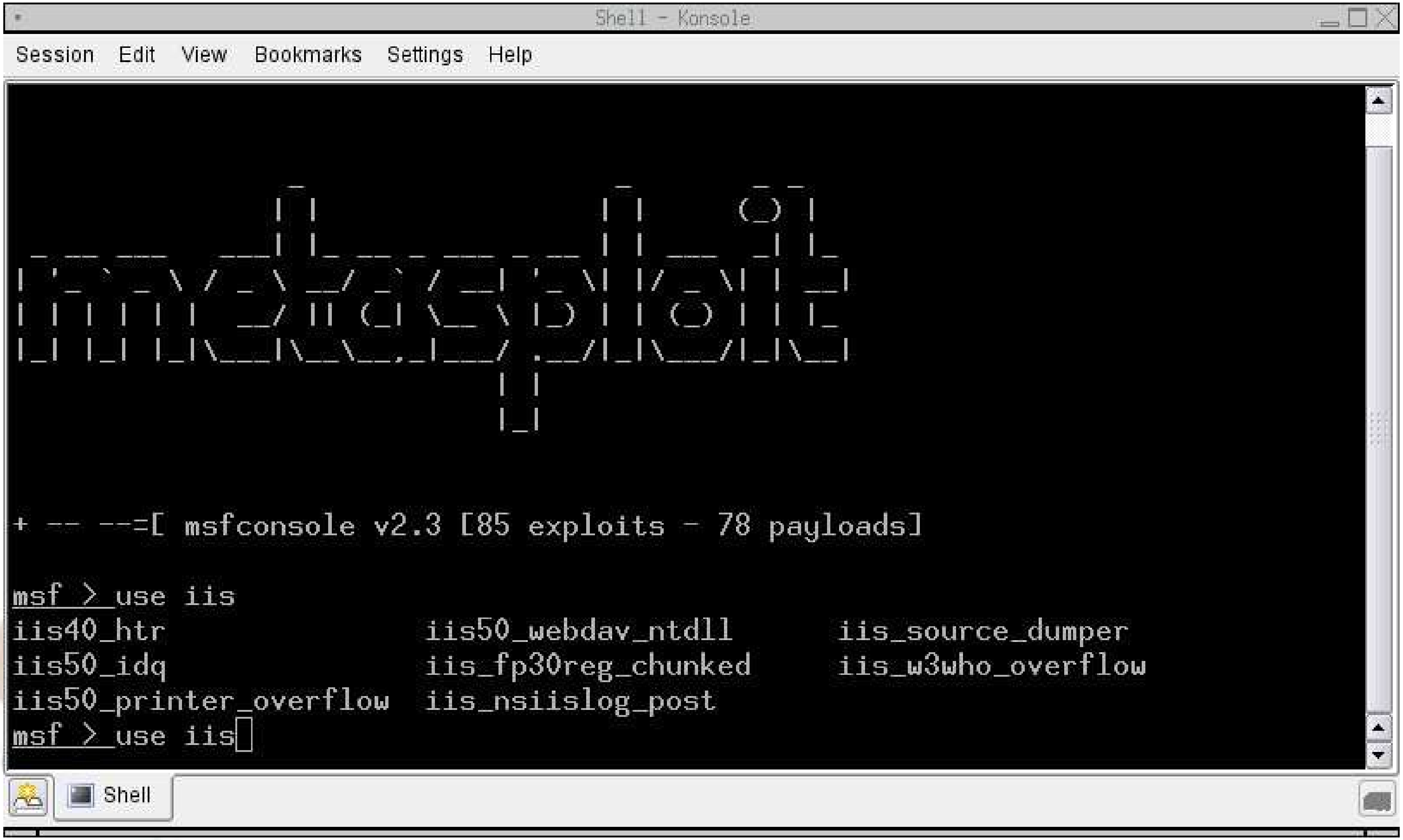
Encoders

- **Even smaller assembly code**
  - Between 15 and 60 bytes long
  - Remove NULL bytes, other bytes
  - XOR-based, additive feedback
  - AlphaNum and unicode support
  - Avoid intrusion detection systems
  - Transparently encode payloads

Nop Generators

- **Instructions that do “nothing”**
  - push, pop, add, sub, xor, mul
  - Nop sleds random by default
- **Multi-byte nop sled generation**
  - OptyNop and OptyNop2
  - Avoid intrusion detection systems

Tab-completion console shell



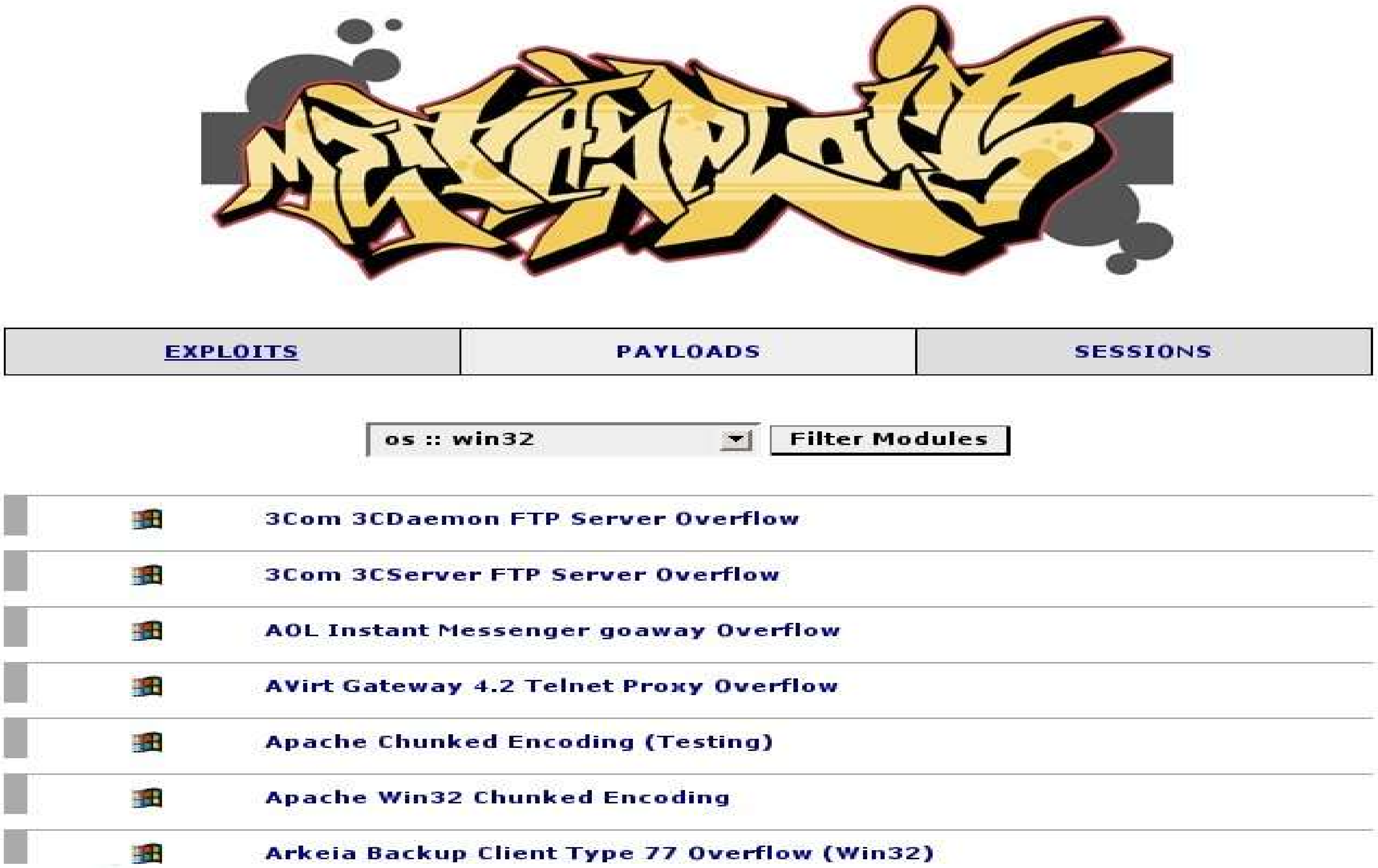
```
                _                  _       _ _
               | |                | |     (_) |
 _ __ ___   ___| |_ __ _ ___ _ __ | | ___  _| |_
| '_ ` _ \ / _ \ __/ _` / __| '_ \| |/ _ \| | __|
| | | | | |  __/ || (_| \__ \ |_) | | (_) | | |_
|_| |_| |_|\___|\__\__,_|___/ .__/|_|\___/|_|\__|
                            | |
                            |_|


+ -- --=[ msfconsole v2.3 [85 exploits - 78 payloads]

msf > use iis
iis40_htr                iis50_webdav_ntdll     iis_source_dumper
iis50_idq                iis_fp30reg_chunked    iis_w3who_overflow
iis50_printer_overflow   iis_nsiislog_post
msf > use iis
```

Web Interface

Click, click, click, shell.



3Com 3CDaemon FTP Server Overflow
3Com 3CServer FTP Server Overflow
AOL Instant Messenger goaway Overflow
AVirt Gateway 4.2 Telnet Proxy Overflow
Apache Chunked Encoding (Testing)
Apache Win32 Chunked Encoding
Arkeia Backup Client Type 77 Overflow (Win32)

Framework Architecture

Modules

- Payloads
- Encoders
- Exploits
- Nops

Interfaces

- Console
- Web
- CLI

Libraries

- Msf
- Pex
- 3rd Party

Core Classes

- Utils
- UI
- Base
- Module

MS 2005

Exploit Process

- Select exploit, show targets
- Select target, show payloads
- Select payload, show options
- Select options, run exploit
  - Encoder tranforms payload
  - Nops pad out the payload
  - Exploit injects encoded payload

Utilities

- **Helper utilities**
  - *msfpescan* » Win32 return addresses
  - *msfelfscan* » Linux return addresses
  - *msfdldebug* » Download symbols
  - *msfpayload* » Generate payloads
  - *msfencode* » Encode payloads
  - *msfupdate* » Online update system

Advanced Payloads

Overview

- **Payloads overview**
  - Tiny little bits of machine code
  - Peform a specific exploit task
  - Bind command shell to a TCP port
  - Send command shell back to attacker
  - Set the **stage** for a bigger payload

Staged Payloads

- **Staged payload systems**
  - Small payload used to load a big one
  - Second stage is sent over network
  - Allows for complex multi-use payloads
  - Useful when payload space is limited
  - Modular payload development

DLL Injection

- **Windows remote DLL injection**
  - A three-stage loading system
  - In-process DLL injection
  - Written by Jarkko and Skape
  - Full access to Windows API
  - Easily convert C/C++ to payload
  - No disk access or new processes :-)

VNC Injection

- Windows VNC server injection
  - Injects VNC server as new thread
  - Reuses existing payload connection
  - Based on RealVNC source code
  - Adapted by Skape and HDM
  - Breaks locked desktops
  - Takes over WinLogon desktop

Meterpreter

- **The Meterpreter**
  - Custom shell written as DLL payload
  - Connection multiplexing (channels)
  - Dynamically load extensions over net
  - Built-in cryptography support
  - Also written by Skape :)

Meterpreter

Meterpreter extensions
  # Execute interactive commands
  # Upload, download, and list files
  # List and terminate processes
  # Integrated TCP port forwarding
  # Dump the SAM password hashes
  # Inject and channel a VNC service

Demonstrations

Questions?

Contact: msfdev@metasploit.com

Code: http://metasploit.com/