GENERAL APPROACH

Set up a controlled, isolated laboratory in which to examine the malware specimen.

Perform behavioral analysis to examine the specimen's interactions with its environment.

Perform static code analysis to further understand the specimen's inner-workings.

Perform dynamic code analysis to understand the more difficult aspects of the code.

If necessary, unpack the specimen.

Repeat steps 2, 3, and 4 (order may vary) until analysis objectives are met.

Document findings and clean-up the laboratory for future analysis. BEHAVIORAL ANALYSIS

IDA PRO FOR STATIC CODE ANALYSIS

Text search	Alt+T
Show strings window	Shift+F12
Show operand as hex value	Q
Insert comment	:
Follow jump or call in view	Enter
Return to previous view	Esc
Go to next view	Ctrl+Enter
Show names window	Shift+F4
Display function's flow chart	F12
Display graph of function calls	Ctrl+F12
Go to program's entry point	Ctrl+E
Go to specific address	G
Rename a variable or function	N
Show listings of name	Ctrl+L
Display listing of segments	Ctrl+S
Show stack of current function	Ctrl+K

BYPASSING MALWARE DEFENSES

To try unpacking quickly, infect the system and dump from memory via LordPE or OllyDump.

For more surgical unpacking, locate the Original Entry Point (OEP) after the unpacker executes.

If you can't unpack cleanly, examine the packed specimen via dynamic code analysis while it runs.

When unpacking in OllyDbg, try SFX (bytewise) and OllyDump's "Find OEP by Section Hop".

Conceal OllyDbg via HideOD and OllyAdvanced.

A JMP or CALL to EAX may indicate the OEP, possibly preceded by POPA or POPAD.

Look out for tricky jumps via SEH, RET, CALL, etc.

If the packer uses SEH, anticipate OEP by tracking stack areas used to store the packers' handlers.

Decode protected data by examining results of the decoding function via dynamic code analysis.

Correct PE header problems with XPELister, LordPE, ImpREC, PEID, etc.

To get closer to OEP, try breaking on unpacker's calls to LoadLibraryA or GetProcAddress.

BEHAVIORAL ANALYSIS

Be ready to revert to good state via dd, VMware snapshots, CoreRestore, Ghost, SteadyState, etc.

Monitor local (Process Monitor, Process Explorer) and network (Wireshark, tcpdump) interactions.

Detect major local changes (RegShot, Autoruns).

Redirect network traffic (hosts file, DNS, Honeyd).

Activate services (IRC, HTTP, SMTP, etc.) as needed to evoke new behavior from the specimen.

OLLYDBG FOR DYNMAIC CODE ANALYSIS

Step into instruction	F7
Step over instruction	F8
Execute till next breakpoint	F9
Execute till next return	Ctrl+F9
Show previous/next executed instruction	- / +
Return to previous view	*
Show memory map	Alt+M
Follow expression in view	Ctrl+G
Insert comment	;
Follow jump or call in view	Enter
Show listings of names	Ctrl+N
New binary search	Ctrl+B
Next binary search result	Ctrl+L
Show listing of software breakpoints	Alt+B
Assemble Instruction	Space
Edit data in memory	Ctrl+E
Show SEH chain	View > SEH chain
Show patches	Ctrl+P

COMMON X86 REGISTERS AND USES

EAX	Addition, Multiplication, Function Results
ECX	Counter
EBP	Base for referencing function arguments and local variables.
ESP	Points to the current "top" of the stack; changes via Push, Pop, and more.
EFLAGS	Contains flags that store outcomes of computations.