GDB-PEDA Cheatsheet – Page 1

Installation

```
# git clone
https://github.com/longld/peda ~/peda
# echo "source ~/peda/peda.py" >>
~/.gdbinit
```

Running

`# gdb <program> [core dump]`
Start GDB (with optional core dump).

`# gdb –args <program> <args…>`
Start GDB and pass arguments.

`# gdb --pid <pid>`
Start GDB and attach to process.

`# gdb <program> -ex <'command'>`
Start GDB and execute command.

`pset|pshow arg <args…>`
Set/show arguments to pass to program to be debugged.

`pset|pshow option <name> <value>`
Set/show various PEDA options.

`pset|pshow env <name> <value>`
Set/show environment variables.

`run`
Run the program to be debugged.

`start`
Start debugged program and stop at most convenient entry.

`kill`
Kill the running program.

Security

`checksec [file]`
Check security options of debugged binary (or target file).

`aslr [on|off]`
Check GDB ASLR setting (or turn it on/off).

`nxtest [address]`
Perform No-Execute (NX) check test.

`unptrace [del]`
Disable/enable anti-ptrace detection.

Miscellaneous

`utils <command> <arg>`
Perform miscellaneous utilities.

`loadmem <file> <address> [size]`
Load content of raw binary file to memory (with optional size).

`session save|restore [filename]`
Save/restore GDB session to/from file.

`snapshot save|restore [filename]`
Save/restore process's snapshot to/from file.

`peda [help command]`
List all PEDA commands and help.

@charleycelice

Information

`elfheader [header_name]`
Get headers information from debugged program.

`readelf <mapname|filename> [header_name]`
Get headers information from target file.

`elfsymbol [symbol_name]`
Get symbol information from debugged program.

`procinfo [pid]`
Fetch information from /proc/pid for debugged program (or optional pid).

`vmmap [address|mapname]`
Get virtual mapping address ranges for debugged process (with optional address/mapname)

`context [reg|code|stack|all] [code/stack length]`
Get current execution context (with optional code/stack length).

`crashdump [reason]`
Get crashdump info (with optional reason text).

`dumpargs [count]`
Get arguments passed to function when stopped at call instruction (with optional display count).

GDB-PEDA Cheatsheet – Page 2

Information, cont.

`dumpmem <file> [<start> <end>|<mapname>]`

Dump content of memory region to file.

`eflags [set|clear|toggle] <flagname>`

Show/set/clear/toggle value of eflags register.

`getfile|getpid`

Get filename/pid of debugged process.

`hexdump|hexprint <address> [count|/count]`

Get hex/ascii or hexified dump of data in memory (with optional count).

`strings [[start] [end]]|[mapname] [minlength]`

Dump strings in memory (with optional start/end addresses, mapname, and minimum length).

`tracecall ["func1,func2"]|["-func1,func2"] [mapname1,mapname2]`

Trace function calls made by the program (with optional specific functions or inverse, and mapname).

`traceinst [count] ["inst1,inst2"] [mapname1,mapname2]`

Trace instructions executed by the program (with optional specific instructions, mapname, and count).

`xinfo <address|register> [reg1 reg2]`

Get information of address/registers.

`xprint <expression>`

Extra support to GDB's print command.

Search

`lookup address|pointer <address> <reg|code|stack|all>`

Search for addresses/references to addresses within memory range.

`searchmem|find <pattern> [[start] [end]|[mapname]]`

Search for patterns in memory (supports regex).

`asmsearch <"expression"> [[start] [end]|[mapname]]`

Search for ASM expression (with optional memory range).

`cmpmem <start> <end> <file>`

Compare content of memory region with file.

`distance <address>|<address1> <address2>`

Calculate distance between address and current stack pointer (or two specified addresses).

`jmpcall ["reg"] [[start] [end]|[mapname]]`

Search for JMP/CALL instructions in memory (with optional range).

`profile [count] [keyword]`

Count executed instructions in the program (with optional count or keyword).

`refsearch <value> [mapname]`

Search all references to a value in memory (with optional range).

`sgrep <pattern> [[start] [end]|[mapname]]`

Search for string patterns (with optional memory range).

`substr <"string"> [[start] [end]|[mapname]]`

Search for substrings in memory (with optional range).

`telescope [address] [linecount]`

Get memory content at an address with smart dereferences.

`xrefs [pattern] [[file]|[mapname]]`

Search for call/data access references to a function/variable.

@charleycelice

GDB-PEDA Cheatsheet – Page 3

Debugging/Patching

`patch <address>|<from_addr> <to_addr> ["string"]`

Patch memory start at an address with string/hexstring/int.

`xormem <start> <end> <key>`

XOR memory region with key.

`deactive <function> [del]`

Deactivate/reactivate function execution in debugged program.

`goto <address>`

Continue execution at an address.

`nextcall|nextjmp [keyword] [mapname1,mapname2]`

Step until next call/jump instruction (with optional keyword and memory range).

`pltbreak [name]`

Set breakpoints at PLT functions (with optional match regex name).

`skipi [count]`

Skip next count of instructions.

`stepuntil <inst1,inst2> [mapname1,mapname2]`

Step until desired instruction (with optional memory range).

`waitfor <cmd> [-c]`

Wait for and attach to specified process (with optional auto continue).

`xuntil <address>|<function>`

Continue execution until address or function.

Dis/Assemble

`pdisass [address] ["gdb disassemble args"]`

GDB disassemble command with colours (and optional address).

`assemble [-b16|-b32|-b64] [address]`

On-the-fly assemble/execute instructions using NASM (with optional mode and address).

`nearpc [address] [count]`

Disassemble instructions near current PC or given address (with optional count).

Exploit Dev.

`shellcode <generate|search|display|zsc>`

Generate/search keywords/display by id/create custom shellcode.

`skeleton <argv|env|stdin|remote> [file]`

Generate python exploit code template.

`payload copybytes [dest1 data1 dest2 data2…]`

Generate ROP payload using ret2plt.

`gennop <size> [chars]`

Generate given length NOP sled (with optional characters set).

`pattern <create|offset|search|patch|arg|env>`

Generate/search/write cyclic pattern to memory.

`dumprop [start end|mapname] [keyword] [depth]`

Dump all ROP gadgets in memory range.

`ropgadget [mapname]`

Get common ROP gadgets of binary or library (with optional range).

`ropsearch <"gadget"> [start end|pagename]`

Search for ROP gadgets (with optional memory range).