Pwntools CLI

Mensi Mohamed Amine

Abstract

Pwntools CLI is a command-line toolkit for quick binary exploitation tasks, built on the Pwntools library. It aids in efficient debugging, analysis, and CTF workflows.

1 Introduction

The Pwntools CLI offers fast, script-free access to exploitation tools like remote connections and ELF inspection, making it ideal for CTFs and reverse engineering.

Contents

2 Pwntools Command-line Tool

Absolutely — here's a **detailed explanation of the Pwntools command-line tool**, including the **official usage message** and a **full breakdown of each subcommand**, complete with examples and context.

2.1 Usage

```
usage: pwn [-h]
           {asm,checksec,constgrep,cyclic,debug,disasm,disablenx,elfdiff,elfpatch,errno,hex,libcdb,phd,pwns
           ↪  trip,scramble,shellcraft,template,unhex,update,version}
           ↪  ...

Pwntools Command-line Interface

positional arguments:
  {asm,checksec,constgrep,cyclic,debug,disasm,disablenx,elfdiff,elfpatch,errno,hex,libcdb,phd,pwnstrip,scra
  ↪  mble,shellcraft,template,unhex,update,version}
    asm                 Assemble shellcode into bytes
    checksec            Check binary security settings
    constgrep           Looking up constants from header files. Example:
                        constgrep -c freebsd -m ^PROT_ '3 + 4'
    cyclic              Cyclic pattern creator/finder
    debug               Debug a binary in GDB
    disasm              Disassemble bytes into text format
    disablenx           Disable NX for an ELF binary
    elfdiff             Compare two ELF files
    elfpatch            Patch an ELF file
    errno               Prints out error messages
    hex                 Hex-encodes data provided on the command line or stdin
    libcdb              Print various information about a libc binary
    phd                 Pretty hex dump
    pwnstrip            Strip binaries for CTF usage
    scramble            Shellcode encoder
    shellcraft          Microwave shellcode -- Easy, fast and delicious
    template            Generate an exploit template
    unhex               Decodes hex-encoded data provided on the command line
                        or via stdin.
    update              Check for pwntools updates
    version             Pwntools version

options:
  -h, --help            show this help message and exit
```

2.2 Detailed Breakdown of Pwntools CLI Tools

Below is an in-depth description of each command shown in the usage above, including **examples**, **use cases**, and **flags**:

2.2.1 'asm': Assemble shellcode into bytes

Convert assembly into machine code (useful for payloads).

Example:

```
pwn asm 'mov eax, 1; int 0x80'
```

Flags:

- '-a' to specify architecture ('i386', 'amd64')
- '-f' output format ('raw', 'hex', 'c')

2.2.2 'checksec': Check binary security settings

Analyze protection mechanisms.
Example:

```
pwn checksec ./binary
```

Outputs:

- RELRO
- Stack canary
- NX
- PIE

2.2.3 'constgrep': Look up constants from header files

Example:

```
pwn constgrep -c linux -m ^PROT_
```

Looks up constants like 'PROT$_{R}EAD$', '$PROT_{E}XEC$', *etc.*

2.2.4 'cyclic': Create or search crash patterns

Useful for buffer overflow offset discovery.
Example:

```
pwn cyclic 200
```

To find crash offset:

```
pwn cyclic -l 0x6161616c
```

2.2.5 'debug': Launch debugger with GEF/PEDA context

Launch 'gdb' with pwntools presets.
Example:

```
pwn debug ./binary
```

2.2.6 'disasm': Disassemble hex into assembly

Convert hex machine code into readable instructions.
Example:

```
pwn disasm 'b801000000cd80'
```

2.2.7 'disablenx': Disable NX on ELF

Make binary stack executable (dangerous!).
Example:

```
pwn disablenx ./binary
```

2.2.8 ‘elfdiff‘: Compare ELF files

Find differences in symbols, headers, etc.

Example:

```
pwn elfdiff binary1 binary2
```

2.2.9 ‘elfpatch‘: Patch bytes into an ELF

Inject values at given addresses.

Example:

```
pwn elfpatch ./binary 0x400123:0x90909090
```

2.2.10 ‘errno‘: Lookup error messages by number or name

Example:

```
pwn errno 13
# → EACCES: Permission denied

pwn errno EINVAL
# → 22: Invalid argument
```

2.2.11 ‘hex‘: Hex-encode data

Turn input into hex format.

Example:

```
pwn hex 'hello'
# → 68656c6c6f
```

2.2.12 ‘unhex‘: Decode hex into plain text

Example:

```
pwn unhex 68656c6c6f
# → hello
```

2.2.13 ‘libcdb‘: Lookup libc versions

Find matching libc from function offsets or versions (CTF tool).

Example:

```
pwn libcdb 0xdeadbeef
```

2.2.14 ‘phd‘: Pretty Hex Dump

Nicer hex output (like ‘xxd‘ but cleaner).
Example:

```
echo 'test' | pwn phd
```

2.2.15 ‘pwnstrip‘: Strip binary for CTF

Removes debug info to reduce binary size.
Example:

```
pwn pwnstrip ./binary
```

2.2.16 ‘scramble‘: Encode shellcode

Obfuscate shellcode (to avoid bad chars).
Example:

```
pwn scramble "$(pwn shellcraft i386.linux.sh)"
```

2.2.17 ‘shellcraft‘: Generate shellcode

Auto-generate shellcode for syscalls or functions.
Example:

```
pwn shellcraft i386.linux.sh
```

To run ‘/bin/sh‘.

2.2.18 ‘template‘: Create exploit boilerplate

Auto-create a pwntools Python script for binary exploitation.
Example:

```
pwn template ./binary
```

2.2.19 ‘update‘: Check for updates

Checks if a new pwntools version is available.

```
pwn update
```

2.2.20 ‘version‘: Show installed pwntools version

```
pwn version
```

2.3 Summary

Command	Purpose
‘asm‘	Assemble shellcode
‘disasm‘	Disassemble bytes
‘checksec‘	View binary protections
‘cyclic‘	Find crash offsets
‘debug‘	Launch GDB
‘disablenx‘	Disable NX in ELF
‘elfpatch‘	Patch binary
‘constgrep‘	Lookup constants from headers
‘errno‘	Decode error codes
‘hex/unhex‘	Convert data to/from hex
‘pwnstrip‘	Strip ELF for CTF
‘scramble‘	Encode shellcode
‘shellcraft‘	Generate shellcode
‘template‘	Make exploit template
‘libcdb‘	Query libc offsets
‘phd‘	Pretty hex dump
‘elfdiff‘	Compare ELF binaries
‘update‘	Update pwntools
‘version‘	Show version