Pretty-printing kernel data structures

lovasko@freebsd.org

AsiaBSDCon 2015
Tokyo, Japan
Why?

- data as the correctness identifier
- tools need to be good with variables
- printf vs. debugger
DDB

- kernel debugger
- FreeBSD, OpenBSD, NetBSD, OS X
- single stepping kernel, breakpoints
- included in GENERIC kernel
- old & lacking some important stuff
The Problem

Enable the kernel debugger DDB to pretty-print data structures that are used in the currently loaded kernel.
Example

```c
struct city {
    unsigned int population;
    char* name;
    float record_high_jan;
}
```
Example

>prettyprint 0x1234 ‘struct city’

0x1234 = struct city {
    unsigned int population = 30
    char* name = “Khartoum”
    float record_high_jan = 39.7
}
Status Quo

• examine command

• can specify content type (string, float, …)

• very important at the low level
Example

>examine/x 0x1124
kdb_sysctl_enter+0x89: 0xc0ffeebab
Better status quo

• show command

• supports few structures: buffer, domain, file, lock
Still not good enough

• still linear approach (code vs. struct)
• what happens when we add a type?
• what happens when we add a member?
• more generic approach!
Need for type information

- format that describes data types
- DWARF (too big though)
- other options?
- we already have CTF
Compact C Type Format (CTF)

- DTrace/mdb origin
- storing types - integers, floats, structs, …
- can represent everything
- very compact
So, how compact?

- BSD kernel
- zfs.ko
- radeonkms.so

Megabytes

CTF

DWARF
The Solution

CTF, duh
Old library

• made at Sun/Joyent
• CDDL license
• unstable private API
• does not contain full implementation
New library

- made by me!
- 2-clause BSD
- FreeBSD support
- github.com/lovasko/libctf
Implementation

• C99 (almost)
• using queue(3)
• using stdint(7)
• unit tests
Average inflation

$ find /boot/kernel -name '*.symbols' -exec ctfmemusage -r {} \; | awk '{s+=$1} END{print s/NR}'

2.67788
Average loading time

- BSD kernel
- zfs.ko
- radeonkms.so
DDB work

- DDB lives in kernel space
- `libctf` needs to adapt (FreeBSD only)
- I/O in the debugger is too limited
- caching the data set before
libctf in every space

- the same codebase shared between kernel and userland

- heavy use of macros

```c
#ifdef _KERNEL
    #define _CTF_FREE(ptr) free(ptr, M_CTF)
#else
    #define _CTF_FREE(ptr) free(ptr)
#endif
```
Casual types

• adapt proper encoding (float as a floating point number, char as a letter, …)

• struct members in offset order with indentation

• optionally follow pointers
Recursive data types

- linked lists, binary trees, n-ary trees, queues, ...
- ubiquitous
- indentation model does not work
Bad example

0x123 = struct two_int_list {
    int a = 11
    int b = 12
    struct two_int_list* next = {
        int a = 21
        int b = 22
        struct two_int_list* next = {
            int a = 31
            int b = 32
            struct two_int_list* next = {
                int a = 41
                int b = 42
                struct two_int_list* next = NULL
            }
        }
    }
}
Good example

\[0x123 = \text{struct two_int_list} = \{
\text{int} \ a = 11 \\
\text{int} \ b = 12
\}
\text{ | next}
\text{v}
\{
\text{int} \ a = 21 \\
\text{int} \ b = 22
\}
\text{ | next}
\text{v}
\text{NULL}\]
mdb approach

- addr::list type field
Detecting data structures

- observe various common patterns
- names, position in the structure, types
- queue(3) and tree(3)
- create a view for each one (or die trying)
Possible views (not done)

- on demand
- interactive tree discovery
- hashtable querying
Frequently asked questions
What about my BSD?

- currently only FreeBSD
- libelf and zlib dependency for libctf
- happy to help
What about my arch?

• currently only x86
• will provide ARM and MIPS port
• happy to help
Why is it not in the tree?

• needs review

• needs testing
Where can I get it?

• will be available at my FreeBSD wiki
• VirtualBox image
What more is there to do?

• finish endianness in libctf
• sizeof command
• finish port to Linux and illumos
Future projects

• CTF userland tools
• libkvm + libctf
• LLVM/Clang integration
• C++
CTF userland tools

- ctfdump
- ctfstats
- ctfcorequery
- ctfmerge
- ctfconvert
- ctfdiff
- ctfmemusage
libkvm + libctf

- utilise this for tools such as ps(1) or netstat(1)
- kernel and userland must not match
- not even architectures
LLVM/Clang integration

- generate CTF data during compilation
- merge CTF data during linking
- new library has good license
C++

• add C++ type support

• needs full circle - DTrace integration
Thanks

• George Neville-Neil
• rest of the FreeBSD Community
• Google
• Foundation and Conference Orgs