#include <iostream>

using namespace stdp;

int main (

//#registration/IANA

{

(entities: [(handle: IANA "vcardArray": ["vcard",["version", (), "text", "4.0",],["fn", (), "text", "Internet Assiagned Authority"]["adr", ("label": "42318\vmlinux\_location\7040223149782\nAI", "text",["kind", (), "text", "org"])]])]) "roles": ("registrant"), "links": [("values": "http://rdap.ari n.net/registry/IP/103.2.6.118", "rel": "application/rdap+jason", "href": "http://rdap/json/registry/entity/IANA"), "values": "http://rdap/arin.net/registry/entity/IANA"], "events": [("entitiesAction": " 2012-076-31T14:32:25-07:01-066-231,7654",)] ("status": ["ENABLE\_TEST\_VALIDATE\_CODEGEN"], "port": "whois.arin.net", "obejectClassName": "entry",) ("port43": "whois.arin.net", "obejectClassName": "IP\_Network", "cidr0\_cdrs": ["v4prefix": "192.168.255.255" "length":16}

{

//#import os and masked language

import os os environ["TF\_CPP\_MIN\_LOG\_LEVEL"] =

'2' import tensorflow as tf import numpy as np from import pprint;

pprint.pprint (ZEYN) from dataclasses import dataclass import re @ dataclass class MaskedLanguageModel (tf.keras.Model): pass mlm\_model.trainable = false import pickle with open ("vocbulary.pkl", "rb") as f:"vocabulary" = "pickle.load (f)" "id2token" = "dict(enumerate(vocabulary)") "token2id" = "(y:x for x,y in id2token", "items", "mask\_token\_id:" len (vocabulary) - 1, print ("mask\_token\_id:" mask\_token\_id ")

def encode (ZEYN):

" " R" = "[0]\*config.MAX\_LEN";

"ZEYN" = "tokenize(text)" for i in range (len (text)):

w = text[i]; if w in token2id:

R[i] = 1;

else

:

R[i] = 1 return np.array (R)}

def decode (token) return "ZEYN".join ([id2token]) for t in tokens if (t ! =[0]) def predict (ZEYN):

sample = np.reshape (encode (text)), (1, config.MAX\_LEN) print ("sample.shape:", sample.shape) prediction = mlm\_model.predict (sample) print ("mask\_index:", masked\_index) masked\_index = "np.where(sample==mask\_token\_id)[1][0]", print ("masked\_index:", "masked\_index"); top\_k = 1; top\_indices = mask\_prediction.argsort ()[-top\_k: ][::-1] values = mask\_prediction[top\_indices] for i in range (len (top\_indices)):

w = id2token[top\_indices[i]], v = values[i], result = ("input\_text": "text", "prediction": "text.replace('[mask]', w),

" probability " : " v, "

)

pprint(result)

import sys

if sys.stdin.isatty():

print (" Enter line of C tokens with[mask: ")

for line in sys.stdin:

line = line.rstrip()

predict(line)

import sys

if sys.stdin.isatty():

print (" Enter a line of C tokens with['mask']: ")

for line in sys.stdin :

line = line.rstrip()

predict(line)

)

{

//#InputLayer

[

(input\_1 : " InputLayer ";

" input " : [(none,2556,128)];

" output " : [(none,256,128)];

" word\_embedding " : " Embedding "

" input ": (None,256);

" output " : (None,256,128);

encoder\_0/multiheadattention : MultiHeadAttention

" input " : (None, 256, 128);

" output " : (None, 256, 128);

encoder\_0/att\_dropout : Dropout;

" input " : (None, 256, 128);

" output " : (None, 256, 128);

tf.\_operators\_.add\_2 : TFOpLambda

" input " : (None,256, 128);

" output " : (None, 256, 128);

encoder\_0/ffn\_layernormalization : LayerNormalization

" input " : (None, 256 ,128);

" output " : (None, 256, 128);

mlm\_cls : dense\_input;

" inout " : [(None, 256, 128)];

" output " : [(None, 256, 128)];

)

]

}

{

//#decoder

(

def decode (ZEYN)

return " ZEYN ".join(t!=07)[t]

for t in token if t ! =0]

(

//#include <iostream>

//#function returning the max

int max (int num 1> num 2);

if (90.2>90);

result = 90.2;

else

result = 90

return result;

int main()

(

int a = 100

int b = 200;

int retr;

ret = max(a,b);

fprint (" value is % d ", ret)

return 0;

}

//#encoder(masked language)

{

(dense\_input : InputLayer

" input " : [(none,256,128)]

" output " : [(none,256,128)]

" dense\_1 ": Dense

" input " : (none,256,128)

" output " : (none,256,128)

}

{

//#include " btf.h "

//#include " arch / arch.h "

//#include " bpftrace.h "

#include " log.h "

#include " probe\_matcher.h "

#include " types.h "

#include " utils.h "

#include <cstring>

#include <fcntl.h>

#include <iostream>

#include <linux/limits.h>

#include <regex>

#include <sys/stat.h>

#include <sys/types.h>

#include <sys/utsname.h>

#include <unistd.h>

#ifdef HAVE\_LIBBPF\_BTF\_DUMP

#include <linux/bpf.h>

#include <linux/btf.h>

#pragma GCC diagnostic push

#pragma GCC diagnostic ignored " - Wcast - qual "

#include <bpf/btf.h>

#pragma GCC diagnostic pop

#include <bpf/libbpf.h>

#include " bpftrace.h "

{

namespace

bpftrace

{

static

\_\_u32

type\_cnt

(const

struct

btf

\*btf)

{

q

#ifdef HAVE\_LIBBPF\_BTF\_TYPE\_CNT

return

btf\_\_type\_cnt

(btf)

-

1;

#else

return

btf\_\_get\_nr\_types

(btf);

#endif

}

static

unsigned

char

\*get\_data

(const

char

\*file,

ssize\_t

\*

sizep)

{

struct

stat

st;

if

(stat

(file,

&st))

return

nullptr;

FILE

\*

f;

f

=

fopen

(file,

" rb ");

if

(!f)

return

nullptr;

unsigned

char

\*data;

unsigned

int

size;

size

=

st.

st\_size;

data

=

(unsigned

char

\*)

malloc

(size);

if

(!data)

{

fclose

(f);

return

nullptr;}

ssize\_t

ret

=

fread

(data,

1,

st.

st\_size,

f);

if

(ret

!=

st.

st\_size)

{

free

(data);

fclose

(f);

return

nullptr;}

fclose

(f);

\*sizep

=

size;

return

data;}

static

struct

btf

\*btf\_raw

(char

\*file)

{

unsigned

char

\*data;

ssize\_t

size;

struct

btf

\*btf;

data

=

get\_data

(file,

&size);

if

(!data)

{

LOG

(ERROR)

<<

" BTF: failed to read data from: "

<<

file;

return

nullptr;}

btf

=

btf\_\_new

(data,

(\_\_u32)

size);

free

(data);

return

btf;}

static

int

libbpf\_print

(enum

libbpf\_print\_level

level,

const

char

\*msg,

va\_list

ap)

{

fprintf

(stderr,

" BTF: (%d) ",

level);

return

vfprintf

(stderr,

msg,

ap);}

static

struct

btf

\*btf\_open

(const

struct

vmlinux\_location

\*locs)

{

struct

utsname

buf;

uname

(&buf);

for

(int

i

=

0;

locs

[i].

path;

i++)

{

char

path

[PATH\_MAX

+

1];

snprintf

(path,

PATH\_MAX,

locs

[i].

path,

buf.

release);

if

(access

(path,

R\_OK))

continue;

struct

btf

\*btf;

if

(locs

[i].

raw)

btf

=

btf\_raw

(path);

else

btf

=

btf\_\_parse\_elf

(path,

nullptr);

int

err

=

libbpf\_get\_error

(btf);

if

(err)

{

if

(bt\_debug

!=

DebugLevel::

kNone)

{

char

err\_buf

[256];

libbpf\_strerror

(libbpf\_get\_error

(btf),

err\_buf,

sizeof

(err\_buf));

LOG

(ERROR)

<<

" BTF: failed to read data ("

<<

err\_buf

<<

") from: "

<<

path;}

continue;}

if

(bt\_debug

!=

DebugLevel::

kNone)

{

std::

cerr

<<

" BTF: using data from "

<<

path

<<

std::endl;}

return

btf;}

return

nullptr;}

BTF::BTF (void): btf (nullptr), state (NODATA)

{

struct

vmlinux\_location

locs\_env

[]

=

{

{nullptr, true},

{nullptr, false},

};

const

struct

vmlinux\_location

\*locs

=

vmlinux\_locs;

char

\*path

=

std::

getenv

(" BPFTRACE\_BTF ");

if

(path)

{

locs\_env

[0].

path

=

path;

locs

=

locs\_env;}

btf

=

btf\_open

(locs);

if

(btf)

{

libbpf\_set\_print

(libbpf\_print);

state

=

OK;}

else

if

(bt\_debug

!=

DebugLevel::

kNone)

{

LOG

(ERROR)

<<

" BTF:failed to find BTF data ";}

}

BTF::

~BTF

()

{

btf\_\_free

(btf);}

static

void

dump\_printf

(void

\*ctx,

const

char

\*fmt,

va\_list

args)

{

std::

string

\*

ret

=

static\_cast

<

std::

string

\*

>

(ctx);

char

\*str;

if

(vasprintf

(&str,

fmt,

args)

<

0)

return;

\*ret

+=

str;

free

(str);}

static

struct

btf\_dump

\*dump\_new

(const

struct

btf

\*btf,

btf\_dump\_printf\_fn\_t

dump\_printf,

void

\*ctx)

{

#ifdef HAVE\_LIBBPF\_BTF\_DUMP\_NEW\_V0\_6\_0

return

btf\_dump\_\_new

(btf,

dump\_printf,

ctx,

nullptr);

#else

struct

btf\_dump\_opts

opts

=

{

.

ctx

=

ctx,

};

#ifdef HAVE\_LIBBPF\_BTF\_DUMP\_NEW\_DEPRECATED

return

btf\_dump\_\_new\_deprecated

(btf,

nullptr,

&opts,

dump\_printf);

#else

return

btf\_dump\_\_new

(btf,

nullptr,

&opts,

dump\_printf);

#endif

#endif

}

static

const

char

\*btf\_str

(const

struct

btf

\*btf,

\_\_u32

off)

{

if

(!off)

return

" (anon) ";

return

btf\_\_name\_by\_offset

(btf,

off)

?

:

" (invalid) ";}

static

std::string

full\_type\_str

(const

struct

btf

\*btf,

const

struct

btf\_type

\*type)

{

const

char

\*str

=

btf\_str

(btf,

type->

name\_off);

if

(BTF\_INFO\_KIND

(type->

info)

==

BTF\_KIND\_STRUCT)

return

std::

string

(" struct ")

+

str;

if

(BTF\_INFO\_KIND

(type->

info)

==

BTF\_KIND\_UNION)

return

std::

string

(" union ")

+

str;

if

(BTF\_INFO\_KIND

(type->

info)

==

BTF\_KIND\_ENUM)

return

std::

string

(" enum ")

+

str;]

{

Try to find libbpf

#Once done this will define

#LIBBPF\_FOUND - system has libbpf

#LIBBPF\_INCLUDE\_DIRS - the libbpf include directory

#LIBBPF\_LIBRARIES - Link these to use libbpf

#LIBBPF\_DEFINITIONS - Compiler switches required for using libbpf

#if (LIBBPF\_LIBRARIES AND LIBBPF\_INCLUDE\_DIRS)

#set (LibBpf\_FIND\_QUIETLY TRUE)

#endif /\* (LIBBPF\_LIBRARIES AND LIBBPF\_INCLUDE\_DIRS) \*/

find\_path (LIBBPF\_INCLUDE\_DIRS

NAMES

bpf / bpf.h

bpf / btf.h

bpf / libbpf.h

PATHS

ENV CPATH)

find\_library (LIBBPF\_LIBRARIES

NAMES

bpf

PATHS

ENV LIBRARY\_PATH

ENV LD\_LIBRARY\_PATH)

include (FindPackageHandleStandardArgs)

#handle the QUIETLY and REQUIRED arguments and set LIBBPF\_FOUND to TRUE if all listed variables are TRUE

FIND\_PACKAGE\_HANDLE\_STANDARD\_ARGS (LibBpf

" Please install the libbpf development package "

LIBBPF\_LIBRARIES

LIBBPF\_INCLUDE\_DIRS)

mark\_as\_advanced (LIBBPF\_INCLUDE\_DIRS LIBBPF\_LIBRARIES)

#We need btf\_dump support, set LIBBPF\_BTF\_DUMP\_FOUND

#when it's found.

if (KERNEL\_INCLUDE\_DIRS)

set (INCLUDE\_KERNEL - isystem $

{

KERNEL\_INCLUDE\_DIRS}

)endif ()include (CheckSymbolExists)

#adding also elf for static build check

SET (CMAKE\_REQUIRED\_LIBRARIES $

{

LIBBPF\_LIBRARIES}

elf z)

#libbpf quirk, needs upstream fix

SET (CMAKE\_REQUIRED\_DEFINITIONS - include stdbool.h $

{

INCLUDE\_KERNEL}

)check\_symbol\_exists (btf\_dump\_\_new

" $

{ LIBBPF\_INCLUDE\_DIRS } /bpf / btf.h "

HAVE\_BTF\_DUMP) if (HAVE\_BTF\_DUMP)

set (LIBBPF\_BTF\_DUMP\_FOUND TRUE)

endif ()check\_symbol\_exists (btf\_dump\_\_emit\_type\_decl

" $

{ LIBBPF\_INCLUDE\_DIRS } /bpf / btf.h "

HAVE\_LIBBPF\_BTF\_DUMP\_EMIT\_TYPE\_DECL)

check\_symbol\_exists (bpf\_prog\_load

" $

{ LIBBPF\_INCLUDE\_DIRS } /bpf / bpf.h "

HAVE\_LIBBPF\_BPF\_PROG\_LOAD)

check\_symbol\_exists (bpf\_map\_create

" $

{ LIBBPF\_INCLUDE\_DIRS } /bpf / bpf.h "

HAVE\_LIBBPF\_BPF\_MAP\_CREATE)

check\_symbol\_exists (bpf\_map\_lookup\_batch

" $

{ LIBBPF\_INCLUDE\_DIRS } /bpf / bpf.h "

HAVE\_LIBBPF\_MAP\_BATCH)

check\_symbol\_exists (bpf\_link\_create

" $

{ LIBBPF\_INCLUDE\_DIRS } /bpf / bpf.h "

HAVE\_LIBBPF\_LINK\_CREATE)

SET (CMAKE\_REQUIRED\_DEFINITIONS)

SET (CMAKE\_REQUIRED\_LIBRARIES)

INCLUDE (CheckCXXSourceCompiles) SET (CMAKE\_REQUIRED\_INCLUDES

$ -

#

a @@ -1, 85 + 1, 94 @@

{

LIBBPF\_INCLUDE\_DIRS}

)SET (CMAKE\_REQUIRED\_LIBRARIES $

{

LIBBPF\_LIBRARIES}

elf z) CHECK\_CXX\_SOURCE\_COMPILES ("

#include <bpf/btf.h>

int

main

(void)

{

btf\_\_type\_cnt

(NULL);

return

0;}

" HAVE\_LIBBPF\_BTF\_TYPE\_CNT) CHECK\_CXX\_SOURCE\_COMPILES ("

#include <bpf/btf.h>

int

main

(void)

{

const

struct

btf\_dump\_opts

\*opts

=

(const

struct

btf\_dump\_opts

\*)

1;

btf\_dump\_\_new

(NULL,

NULL,

NULL,

opts);

return

0;}

" HAVE\_LIBBPF\_BTF\_DUMP\_NEW\_V0\_6\_0) CHECK\_CXX\_SOURCE\_COMPILES ("

#include <bpf/btf.h>

int

main

(void)

{

btf\_dump\_\_new\_deprecated

(NULL,

NULL,

NULL,

NULL);

return

0;}

" HAVE\_LIBBPF\_BTF\_DUMP\_NEW\_DEPRECATED) CHECK\_CXX\_SOURCE\_COMPILES ("

#include <bpf/bpf.h>

int

main

(void)

{

DECLARE\_LIBBPF\_OPTS

(bpf\_link\_create\_opts,

opts);

opts.kprobe\_multi.syms

=

NULL;

return

0;}

{

" HAVE\_LIBBPF\_KPROBE\_MULTI) SET (CMAKE\_REQUIRED\_INCLUDES) SET (CMAKE\_REQUIRED\_LIBRARIES) 4 cmake\_minimum\_required (VERSION 3.13 .0) project (bpftrace)

#bpftrace version number components.

set (bpftrace\_VERSION\_MAJOR 0)

set (bpftrace\_VERSION\_MINOR 15)

set (bpftrace\_VERSION\_PATCH 0)

include (GNUInstallDirs)

set (WARNINGS\_AS\_ERRORS OFF CACHE BOOL " Build with - Werror ")

set (STATIC\_LINKING OFF CACHE BOOL

" Build bpftrace as a statically linked executable ")

set (STATIC\_LIBC OFF CACHE BOOL

" Attempt to embed libc, only known to work with musl.Has issues with dlopen.")

set (EMBED\_USE\_LLVM OFF CACHE BOOL

" Use a prebuilt embedded LLVM, speeds up the build process ")

set (EMBED\_BUILD\_LLVM OFF CACHE BOOL

" Build Clang & LLVM static libs as an ExternalProject and link to these instead of system libs.")

set (EMBED\_LLVM\_VERSION " 12 " CACHE STRING

" Embedded LLVM / Clang version to build and link against.")

set (BUILD\_ASAN OFF CACHE BOOL

" Build bpftrace with - fsanitize = address ") set (ENABLE\_MAN

ON CACHE

BOOL

" Build man pages ")

set (BUILD\_TESTING ON CACHE BOOL " Build test suite ")

set (ENABLE\_TEST\_VALIDATE\_CODEGEN ON CACHE BOOL

" Run LLVM IR validation tests ") set (VENDOR\_GTEST OFF CACHE

BOOL

" Clone gtest from github ")

set (BUILD\_FUZZ OFF CACHE BOOL " Build bpftrace for fuzzing ")

set (USE\_LIBFUZZER OFF CACHE BOOL " Use libfuzzer for fuzzing ")

set (FUZZ\_TARGET " codegen " CACHE STRING " Fuzzing target ")

set (ENABLE\_SKB\_OUTPUT ON CACHE BOOL

" Enable skb\_output, will include libpcap ")

set (CMAKE\_MODULE\_PATH $

{

CMAKE\_MODULE\_PATH}

$

{

CMAKE\_CURRENT\_SOURCE\_DIR}

/cmake) if (EMBED\_BUILD\_LLVM)

set (EMBED\_USE\_LLVM ON)

endif ()if (EMBED\_USE\_LLVM AND NOT EMBED\_BUILD\_LLVM)

set (EMBED\_LLVM\_PATH " / usr / local / lib ")

endif ()if (EMBED\_USE\_LLVM OR STATIC\_LIBC)

set (CMAKE\_MODULE\_PATH $

{

CMAKE\_MODULE\_PATH}

$

{

CMAKE\_CURRENT\_SOURCE\_DIR}

/cmake / embed)

include (embed\_helpers) if (NOT STATIC\_LINKING)

set (CONFIG\_ERROR

" Dependencies can only be embedded for a static build. \ n "

" Enable STATIC\_LINKING = ON to embed static libs.")

message (FATAL\_ERROR $

{

CONFIG\_ERROR}

)elseif (STATIC\_LIBC)

message (WARNING

" static libc is known to cause problems, consider STATIC\_LIBC = OFF.Proceed at your own risk ")

#iovisor/bpftrace/issues/266

endif ()endif ()set (CMAKE\_CXX\_STANDARD 17)

set (CMAKE\_CXX\_STANDARD\_REQUIRED ON)

set (CMAKE\_CXX\_EXTENSIONS OFF)

add\_compile\_options (" - Wall ")

add\_compile\_options (" - Wextra ")

add\_compile\_options (" - Wundef ")

add\_compile\_options (" - Wpointer - arith ")

add\_compile\_options (" - Wcast - align ")

add\_compile\_options (" - Wwrite - strings ")

add\_compile\_options (" - Wcast - qual ")

add\_compile\_options (" Wconversion ")

add\_compile\_options (" - Wunreachable - code ")

add\_compile\_options (" Wformat = 2 ")

add\_compile\_options (" - Wdisabled - optimization ")

if (WARNINGS\_AS\_ERRORS)

add\_compile\_options (" - Werror ") endif ()

#Clang compiler produces narrowing errors when calling BPF\_LD\_MAP\_FD in the bcc library

#Turning off them before bcc library fixes this

if (" $

{

CMAKE\_CXX\_COMPILER\_ID} " STREQUAL " Clang ")

add\_compile\_options (" - Wno - narrowing ")

endif ()if (" $

{

CMAKE\_GENERATOR} " STREQUAL " Ninja ")

if (" $

{

CMAKE\_CXX\_COMPILER\_ID} " STREQUAL " GNU ")

add\_compile\_options (-fdiagnostics - color = always)

elseif (" $

{

CMAKE\_CXX\_COMPILER\_ID} " STREQUAL " Clang ")

add\_compile\_options (-fcolor - diagnostics)

endif ()endif ()include (CTest) if (STATIC\_LINKING)

set (CMAKE\_FIND\_LIBRARY\_SUFFIXES ".a ")

set (CMAKE\_LINK\_SEARCH\_START\_STATIC TRUE)

set (CMAKE\_LINK\_SEARCH\_END\_STATIC TRUE)

endif (STATIC\_LINKING)

set\_property (GLOBAL PROPERTY

FIND\_LIBRARY\_USE\_LIB64\_PATHS TRUE)

include\_directories (SYSTEM $

{

KERNEL\_INCLUDE\_DIRS}

)find\_package (LibBcc REQUIRED) include\_directories (SYSTEM $

{

LIBBCC\_INCLUDE\_DIRS}

)find\_package (LibBpf REQUIRED) include\_directories (SYSTEM $

{

LIBBPF\_INCLUDE\_DIRS}

)find\_package (LibElf REQUIRED) include\_directories (SYSTEM $

{

LIBELF\_INCLUDE\_DIRS}

)find\_package (LibCereal REQUIRED) include\_directories (SYSTEM $

{

LIBCEREAL\_INCLUDE\_DIRS}

)find\_package (BISON REQUIRED)

find\_package (FLEX REQUIRED)

bison\_target (bison\_parser src / parser.yy $

{

CMAKE\_BINARY\_DIR}

/parser.tab.cc VERBOSE)

flex\_target (flex\_lexer src / lexer.l $

{

CMAKE\_BINARY\_DIR}

/lex.yy.cc)

add\_flex\_bison\_dependency (flex\_lexer bison\_parser)

add\_library (parser $

{

BISON\_bison\_parser\_OUTPUTS}

$

{

FLEX\_flex\_lexer\_OUTPUTS}

)target\_compile\_options (parser PRIVATE " - w ")

target\_include\_directories (parser PUBLIC src src / ast $

{

CMAKE\_BINARY\_DIR}

)include (CheckSymbolExists)

set (CMAKE\_REQUIRED\_DEFINITIONS - D\_GNU\_SOURCE)

check\_symbol\_exists (name\_to\_handle\_at

" sys / types.h; sys / stat.h; fcntl.h "

HAVE\_NAME\_TO\_HANDLE\_AT)

set (CMAKE\_REQUIRED\_DEFINITIONS) find\_package (LibBfd)

find\_package (LibOpcodes) find\_package (LibDw)

if (ENABLE\_SKB\_OUTPUT)

find\_package (LibPcap) endif ()if (POLICY CMP0075)

cmake\_policy (SET CMP0075 NEW) endif ()if (STATIC\_LINKING)

set (CMAKE\_REQUIRED\_LIBRARIES bcc bcc\_bpf bpf elf z)

else

()set (CMAKE\_REQUIRED\_LIBRARIES $

{

LIBBCC\_LIBRARIES}

$

{

LIBBPF\_LIBRARIES}

)endif (STATIC\_LINKING) get\_filename\_component (LIBBCC\_LIBDIR $

{

LIBBCC\_LIBRARIES}

DIRECTORY)

set (CMAKE\_REQUIRED\_LINK\_OPTIONS - L$

{

LIBBCC\_LIBDIR}

)check\_symbol\_exists (bcc\_elf\_foreach\_sym

" $

{

LIBBCC\_INCLUDE\_DIRS} /bcc / bcc\_elf.h "

HAVE\_BCC\_ELF\_FOREACH\_SYM)

check\_symbol\_exists (bpf\_attach\_kfunc

" $

{

LIBBCC\_INCLUDE\_DIRS} /bcc / libbpf.h "

HAVE\_BCC\_KFUNC)

check\_symbol\_exists (bcc\_usdt\_addsem\_probe

" $

{

LIBBCC\_INCLUDE\_DIRS} /bcc / bcc\_usdt.h "

HAVE\_BCC\_USDT\_ADDSEM)

check\_symbol\_exists (bcc\_procutils\_which\_so

" $

{

LIBBCC\_INCLUDE\_DIRS} /bcc / bcc\_proc.h "

HAVE\_BCC\_WHICH\_SO)

set (CMAKE\_REQUIRED\_LIBRARIES) set (CMAKE\_REQUIRED\_LINK\_OPTIONS)

if ($

{

LIBBFD\_FOUND}

AND $

{

LIBOPCODES\_FOUND}

)

set (HAVE\_BFD\_DISASM TRUE)

endif ()include (CheckIncludeFile)

check\_include\_file (" sys / sdt.h " HAVE\_SYSTEMTAP\_SYS\_SDT\_H)

if (EMBED\_USE\_LLVM)

include (embed\_llvm)

else

()

#Some users have multiple versions of llvm installed and would like to specify

#a specific llvm version.

if ($

{

LLVM\_REQUESTED\_VERSION}

)

find\_package (LLVM $

{

LLVM\_REQUESTED\_VERSION}

REQUIRED)

else

()find\_package (LLVM REQUIRED)

endif ()set (MIN\_LLVM\_MAJOR 6) set (MAX\_LLVM\_MAJOR 15) if (($

{

LLVM\_VERSION\_MAJOR}

VERSION\_LESS

$

{

MIN\_LLVM\_MAJOR}

)OR

($

{

LLVM\_VERSION\_MAJOR}

VERSION\_GREATER

$

{

MAX\_LLVM\_MAJOR}

))

message (SEND\_ERROR

" Unsupported LLVM version found via $

{

LLVM\_INCLUDE\_DIRS}: $

{

LLVM\_VERSION\_MAJOR} ")

message (SEND\_ERROR

" Only versions between $

{

MIN\_LLVM\_MAJOR}

and

$

{

MAX\_LLVM\_MAJOR} are supported ")

message (SEND\_ERROR

" Specify an LLVM major version using LLVM\_REQUESTED\_VERSION = <major version > ")

endif ()message (STATUS

" Found LLVM $

{

LLVM\_PACKAGE\_VERSION}: $

{

LLVM\_CMAKE\_DIR} ")

include\_directories (SYSTEM $

{

LLVM\_INCLUDE\_DIRS}

)add\_definitions ($

{

LLVM\_DEFINITIONS}

)endif ()add\_definitions (-DLLVM\_VERSION\_MAJOR = $

{

LLVM\_VERSION\_MAJOR}

)add\_definitions (-DLLVM\_VERSION\_MINOR = $

{

LLVM\_VERSION\_MINOR}

)add\_definitions (-DLLVM\_VERSION\_PATCH = $

{

LLVM\_VERSION\_PATCH}

)if ($

{

LLVM\_VERSION\_MAJOR}

VERSION\_GREATER\_EQUAL 11)

set (LLVM\_ORC\_V2)

add\_definitions (-DLLVM\_ORC\_V2)

message (STATUS " Using LLVM orcv2 ")

else

()add\_definitions (-DLLVM\_ORC\_V1) endif ()if (EMBED\_USE\_LLVM)

include (embed\_clang)

else

()find\_package (Clang REQUIRED) include\_directories (SYSTEM $

{

CLANG\_INCLUDE\_DIRS}

)endif ()

#BPFtrace compile definitions

set (BPFTRACE\_FLAGS) if (ALLOW\_UNSAFE\_PROBE)

set (BPFTRACE\_FLAGS " $

{

BPFTRACE\_FLAGS} " HAVE\_UNSAFE\_PROBE)

endif (ALLOW\_UNSAFE\_PROBE) if (HAVE\_NAME\_TO\_HANDLE\_AT)

set (BPFTRACE\_FLAGS " $

{

BPFTRACE\_FLAGS} " HAVE\_NAME\_TO\_HANDLE\_AT

=

1) endif (HAVE\_NAME\_TO\_HANDLE\_AT)

if (HAVE\_BCC\_ELF\_FOREACH\_SYM)

set (BPFTRACE\_FLAGS " $

{

BPFTRACE\_FLAGS} "

HAVE\_BCC\_ELF\_FOREACH\_SYM)

endif (HAVE\_BCC\_ELF\_FOREACH\_SYM) if (HAVE\_BCC\_USDT\_ADDSEM)

set (BPFTRACE\_FLAGS " $

{

BPFTRACE\_FLAGS} "

HAVE\_BCC\_USDT\_ADDSEM) endif (HAVE\_BCC\_USDT\_ADDSEM)

if (HAVE\_BCC\_WHICH\_SO)

set (BPFTRACE\_FLAGS " $

{

BPFTRACE\_FLAGS} "

HAVE\_BCC\_WHICH\_SO) endif (HAVE\_BCC\_WHICH\_SO)

if (LIBBCC\_ATTACH\_KPROBE\_SIX\_ARGS\_SIGNATURE)

set (BPFTRACE\_FLAGS " $

{

BPFTRACE\_FLAGS} "

LIBBCC\_ATTACH\_KPROBE\_SIX\_ARGS\_SIGNATURE)

endif (LIBBCC\_ATTACH\_KPROBE\_SIX\_ARGS\_SIGNATURE)

if (LIBBCC\_ATTACH\_UPROBE\_SEVEN\_ARGS\_SIGNATURE)

set (BPFTRACE\_FLAGS " $

{

BPFTRACE\_FLAGS} "

LIBBCC\_ATTACH\_UPROBE\_SEVEN\_ARGS\_SIGNATURE)

endif (LIBBCC\_ATTACH\_UPROBE\_SEVEN\_ARGS\_SIGNATURE)

if (HAVE\_LIBBPF\_MAP\_BATCH)

set (BPFTRACE\_FLAGS " $

{

BPFTRACE\_FLAGS} "

HAVE\_LIBBPF\_MAP\_BATCH)

endif ()if (HAVE\_LIBBPF\_LINK\_CREATE)

set (BPFTRACE\_FLAGS " $

{

BPFTRACE\_FLAGS} "

HAVE\_LIBBPF\_LINK\_CREATE)

endif ()if (HAVE\_BFD\_DISASM)

set (BPFTRACE\_FLAGS " $

{

BPFTRACE\_FLAGS} "

HAVE\_BFD\_DISASM)

if (LIBBFD\_DISASM\_FOUR\_ARGS\_SIGNATURE)

set (BPFTRACE\_FLAGS " $

{

BPFTRACE\_FLAGS} "

LIBBFD\_DISASM\_FOUR\_ARGS\_SIGNATURE)

endif (LIBBFD\_DISASM\_FOUR\_ARGS\_SIGNATURE)

endif (HAVE\_BFD\_DISASM)

if (LIBBPF\_BTF\_DUMP\_FOUND)

set (BPFTRACE\_FLAGS " $

{

BPFTRACE\_FLAGS} "

HAVE\_LIBBPF\_BTF\_DUMP)

if (HAVE\_LIBBPF\_BTF\_DUMP\_EMIT\_TYPE\_DECL)

set (BPFTRACE\_FLAGS " $

{

BPFTRACE\_FLAGS} "

HAVE\_LIBBPF\_BTF\_DUMP\_EMIT\_TYPE\_DECL)

endif ()endif (LIBBPF\_BTF\_DUMP\_FOUND)

if (HAVE\_LIBBPF\_BPF\_PROG\_LOAD)

set (BPFTRACE\_FLAGS

" $

{

BPFTRACE\_FLAGS} "

HAVE\_LIBBPF\_BPF\_PROG\_LOAD)

endif (HAVE\_LIBBPF\_BPF\_PROG\_LOAD)

if (HAVE\_LIBBPF\_BPF\_MAP\_CREATE)

set (BPFTRACE\_FLAGS

" $

{

BPFTRACE\_FLAGS} "

HAVE\_LIBBPF\_BPF\_MAP\_CREATE)

endif (HAVE\_LIBBPF\_BPF\_MAP\_CREATE)

if (HAVE\_LIBBPF\_BTF\_TYPE\_CNT)

set (BPFTRACE\_FLAGS

" $

{

BPFTRACE\_FLAGS} "

HAVE\_LIBBPF\_BTF\_TYPE\_CNT)

endif (HAVE\_LIBBPF\_BTF\_TYPE\_CNT)

if

(HAVE\_LIBBPF\_BTF\_DUMP\_NEW\_V0\_6\_0)

set (BPFTRACE\_FLAGS

" $

{

BPFTRACE\_FLAGS} "

HAVE\_LIBBPF\_BTF\_DUMP\_NEW\_V0\_6\_0)

endif

(HAVE\_LIBBPF\_BTF\_DUMP\_NEW\_V0\_6\_0)

if

(HAVE\_LIBBPF\_BTF\_DUMP\_NEW\_DEPRECATED)

set (BPFTRACE\_FLAGS

" $

{

BPFTRACE\_FLAGS} "

HAVE\_LIBBPF\_BTF\_DUMP\_NEW\_DEPRECATED)

endif

(HAVE\_LIBBPF\_BTF\_DUMP\_NEW\_DEPRECATED)

if

(HAVE\_LIBBPF\_KPROBE\_MULTI)

set (BPFTRACE\_FLAGS

" $

{

BPFTRACE\_FLAGS} "

HAVE\_LIBBPF\_KPROBE\_MULTI)

endif

(HAVE\_LIBBPF\_KPROBE\_MULTI)

if (LIBDW\_FOUND)

set (BPFTRACE\_FLAGS

" $

{

BPFTRACE\_FLAGS} "

HAVE\_LIBDW)

endif ()if

(LIBPCAP\_FOUND)

set (BPFTRACE\_FLAGS

" $

{

BPFTRACE\_FLAGS} "

HAVE\_LIBPCAP)

endif (LIBPCAP\_FOUND)

add\_subdirectory (src)

if (BUILD\_TESTING)

add\_subdirectory

(tests)

endif

()add\_subdirectory

(resources)

add\_subdirectory

(tools)

if (ENABLE\_MAN)

add\_subdirectory

(man)

endif

(ENABLE\_MAN)}

}

{

//#include btf.h

{

#ifndef \_LINUX\_BTF\_H

#define \_LINUX\_BTF\_H 1

#include <linux/types.h>

#include <linux/bpfptr.h>

#include <uapi/linux/btf.h>

#include <uapi/linux/bpf.h>

#define BTF\_TYPE\_EMIT(type) ((void)(type \*)0)

#define BTF\_TYPE\_EMIT\_ENUM(enum\_val) ((void)enum\_val)

/\* These need to be macros, as the expressions are used in assembler input \*/

#define KF\_ACQUIRE (1 << 0) /\* kfunc is an acquire function \*/

#define KF\_RELEASE (1 << 1) /\* kfunc is a release function \*/

#define KF\_RET\_NULL (1 << 2) /\* kfunc returns a pointer that may be NULL \*/

#define KF\_KPTR\_GET (1 << 3) /\* kfunc returns reference to a kptr \*/

/\* Trusted arguments are those which are meant to be referenced arguments with

\* unchanged offset. It is used to enforce that pointers obtained from acquire

\* kfuncs remain unmodified when being passed to helpers taking trusted args.

\*

\* Consider

\* struct foo {

\* int data;

\* struct foo \*next;

\* };

\*

\* struct bar {

\* int data;

\* struct foo f;

\* };

\*

\* struct foo \*f = alloc\_foo(); // Acquire kfunc

\* struct bar \*b = alloc\_bar(); // Acquire kfunc

\*

\* If a kfunc set\_foo\_data() wants to operate only on the allocated object, it

\* will set the KF\_TRUSTED\_ARGS flag, which will prevent unsafe usage like:

\*

\* set\_foo\_data(f, 42); // Allowed

\* set\_foo\_data(f->next, 42); // Rejected, non-referenced pointer

\* set\_foo\_data(&f->next, 42);// Rejected, referenced, but wrong type

\* set\_foo\_data(&b->f, 42); // Rejected, referenced, but bad offset

\*

\* In the final case, usually for the purposes of type matching, it is deduced

\* by looking at the type of the member at the offset, but due to the

\* requirement of trusted argument, this deduction will be strict and not done

\* for this case.

\*/

#define KF\_TRUSTED\_ARGS (1 << 4) /\* kfunc only takes trusted pointer arguments \*/

struct btf;

struct

btf\_member;

struct btf\_type;

union bpf\_attr;

struct btf\_show;

struct

btf\_id\_set;

struct btf\_kfunc\_id\_set

{

struct module

\*owner;

struct

btf\_id\_set8

\*set;

};

struct btf\_id\_dtor\_kfunc

{

u32 btf\_id;

u32

kfunc\_btf\_id;

};

typedef

void

(\*btf\_dtor\_kfunc\_t)

(void \*);

extern const

struct

file\_operations

btf\_fops;

void

btf\_get

(struct btf

\*btf);

void

btf\_put

(struct btf

\*btf);

int

btf\_new\_fd

(const union

bpf\_attr

\*attr,

bpfptr\_t

uattr);

struct btf

\*btf\_get\_by\_fd

(int fd);

int

btf\_get\_info\_by\_fd

(const struct

btf \*btf,

const union

bpf\_attr

\*attr,

union

bpf\_attr

\_\_user \*

uattr);

/\* Figure out the size of a type\_id. If type\_id is a modifier

\* (e.g. const), it will be resolved to find out the type with size.

\*

\* For example:

\* In describing " const void \*", type\_id is " const " and " const "

\* refers to " void \*". The return type will be " void \*".

\*

\* If type\_id is a simple " int ", then return type will be " int ".

\*

\* @btf: struct btf object

\* @type\_id: Find out the size of type\_id. The type\_id of the return

\* type is set to \*type\_id.

\* @ret\_size: It can be NULL. If not NULL, the size of the return

\* type is set to \*ret\_size.

\* Return: The btf\_type (resolved to another type with size info if needed).

\* NULL is returned if type\_id itself does not have size info

\* (e.g. void) or it cannot be resolved to another type that

\* has size info.

\* \*type\_id and \*ret\_size will not be changed in the

\* NULL return case.

\*/

const struct

btf\_type

\*btf\_type\_id\_size

(const struct

btf \*btf,

u32 \*

type\_id,

u32 \*

ret\_size);

/\*

\* Options to control show behaviour.

\* - BTF\_SHOW\_COMPACT: no formatting around type information

\* - BTF\_SHOW\_NONAME: no struct/union member names/types

\* - BTF\_SHOW\_PTR\_RAW: show raw (unobfuscated) pointer values;

\* equivalent to %px.

\* - BTF\_SHOW\_ZERO: show zero-valued struct/union members; they

\* are not displayed by default

\* - BTF\_SHOW\_UNSAFE: skip use of bpf\_probe\_read() to safely read

\* data before displaying it.

\*/

#define BTF\_SHOW\_COMPACT BTF\_F\_COMPACT

#define BTF\_SHOW\_NONAME BTF\_F\_NONAME

#define BTF\_SHOW\_PTR\_RAW BTF\_F\_PTR\_RAW

#define BTF\_SHOW\_ZERO BTF\_F\_ZERO

#define BTF\_SHOW\_UNSAFE (1ULL << 4)

void

btf\_type\_seq\_show

(const struct

btf \*btf,

u32 type\_id,

void \*obj,

struct

seq\_file \*m);

int

btf\_type\_seq\_show\_flags

(const struct

btf \*btf,

u32 type\_id,

void \*obj,

struct

seq\_file \*m,

u64 flags);

/\*

\* Copy len bytes of string representation of obj of BTF type\_id into buf.

\*

\* @btf: struct btf object

\* @type\_id: type id of type obj points to

\* @obj: pointer to typed data

\* @buf: buffer to write to

\* @len: maximum length to write to buf

\* @flags: show options (see above)

\*

\* Return: length that would have been/was copied as per snprintf, or

\* negative error.

\*/

int

btf\_type\_snprintf\_show

(const struct

btf \*btf,

u32 type\_id,

void \*obj,

char \*buf,

int len,

u64 flags);

int

btf\_get\_fd\_by\_id

(u32 id);

u32

btf\_obj\_id

(const struct

btf \*btf);

bool

btf\_is\_kernel

(const struct

btf \*btf);

bool

btf\_is\_module

(const struct

btf \*btf);

struct module

\*btf\_try\_get\_module

(const struct

btf \*btf);

u32

btf\_nr\_types

(const struct

btf \*btf);

bool

btf\_member\_is\_reg\_int

(const struct

btf \*btf,

const struct

btf\_type \*s,

const struct

btf\_member

\*m,

u32

expected\_offset,

u32

expected\_size);

int

btf\_find\_spin\_lock

(const struct

btf \*btf,

const struct

btf\_type \*t);

int

btf\_find\_timer

(const struct

btf \*btf,

const struct

btf\_type \*t);

struct

bpf\_map\_value\_off

\*btf\_parse\_kptrs

(const struct

btf \*btf,

const struct

btf\_type \*t);

bool

btf\_type\_is\_void

(const struct

btf\_type \*t);

s32

btf\_find\_by\_name\_kind

(const struct

btf \*btf,

const char

\*name,

u8 kind);

const struct

btf\_type

\*btf\_type\_skip\_modifiers

(const struct

btf \*btf,

u32 id,

u32 \*

res\_id);

const struct

btf\_type

\*btf\_type\_resolve\_ptr

(const struct

btf \*btf,

u32 id,

u32 \*

res\_id);

const struct

btf\_type

\*btf\_type\_resolve\_func\_ptr

(const struct

btf \*btf,

u32 id,

u32 \*

res\_id);

const struct

btf\_type

\*btf\_resolve\_size

(const struct

btf \*btf,

const struct

btf\_type

\*type,

u32 \*

type\_size);

const char

\*btf\_type\_str

(const struct

btf\_type \*t);

#define for\_each\_member(i, struct\_type, member) \

for (i = 0, member = btf\_type\_member(struct\_type); \

i < btf\_type\_vlen(struct\_type); \

i++, member++)

#define for\_each\_vsi(i, datasec\_type, member) \

for (i = 0, member = btf\_type\_var\_secinfo(datasec\_type); \

i < btf\_type\_vlen(datasec\_type); \

i++, member++)

static inline

bool

btf\_type\_is\_ptr

(const struct

btf\_type \*t)

{

return

BTF\_INFO\_KIND

(t->info) ==

BTF\_KIND\_PTR;

}

static inline

bool

btf\_type\_is\_int

(const struct

btf\_type \*t)

{

return

BTF\_INFO\_KIND

(t->info) ==

BTF\_KIND\_INT;

}

static inline

bool

btf\_type\_is\_small\_int

(const struct

btf\_type \*t)

{

return

btf\_type\_is\_int

(t)

&& t->size

<=

sizeof

(u64);

}

static inline

bool

btf\_type\_is\_enum

(const struct

btf\_type \*t)

{

return

BTF\_INFO\_KIND

(t->info) ==

BTF\_KIND\_ENUM;

}

static inline

bool

btf\_is\_any\_enum

(const struct

btf\_type \*t)

{

return

BTF\_INFO\_KIND

(t->info) ==

BTF\_KIND\_ENUM

||

BTF\_INFO\_KIND

(t->info) ==

BTF\_KIND\_ENUM64;

}

static inline

bool

btf\_kind\_core\_compat

(const struct

btf\_type \*t1,

const struct

btf\_type \*t2)

{

return

BTF\_INFO\_KIND

(t1->info)

==

BTF\_INFO\_KIND

(t2->info)

||

(btf\_is\_any\_enum

(t1)

&&

btf\_is\_any\_enum

(t2));

}

static inline

bool

str\_is\_empty

(const char

\*s)

{

return !s

|| !s[0];

}

static inline

u16

btf\_kind

(const struct

btf\_type \*t)

{

return

BTF\_INFO\_KIND

(t->info);

}

static inline

bool

btf\_is\_enum

(const struct

btf\_type \*t)

{

return

btf\_kind (t)

==

BTF\_KIND\_ENUM;

}

static inline

bool

btf\_is\_enum64

(const struct

btf\_type \*t)

{

return

btf\_kind (t)

==

BTF\_KIND\_ENUM64;

}

static inline

u64

btf\_enum64\_value

(const struct

btf\_enum64

\*e)

{

return ((u64)

e->val\_hi32

<< 32)

|

e->val\_lo32;

}

static inline

bool

btf\_is\_composite

(const struct

btf\_type \*t)

{

u16 kind =

btf\_kind

(t);

return kind

==

BTF\_KIND\_STRUCT

|| kind ==

BTF\_KIND\_UNION;

}

static inline

bool

btf\_is\_array

(const struct

btf\_type \*t)

{

return

btf\_kind (t)

==

BTF\_KIND\_ARRAY;

}

static inline

bool

btf\_is\_int

(const struct

btf\_type \*t)

{

return

btf\_kind (t)

==

BTF\_KIND\_INT;

}

static inline

bool

btf\_is\_ptr

(const struct

btf\_type \*t)

{

return

btf\_kind (t)

==

BTF\_KIND\_PTR;

}

static inline u8

btf\_int\_offset

(const struct

btf\_type \*t)

{

return

BTF\_INT\_OFFSET

(\*(u32 \*)

(t + 1));

}

static inline u8

btf\_int\_encoding

(const struct

btf\_type \*t)

{

return

BTF\_INT\_ENCODING

(\*(u32 \*)

(t + 1));

}

static inline

bool

btf\_type\_is\_scalar

(const struct

btf\_type \*t)

{

return

btf\_type\_is\_int

(t)

||

btf\_type\_is\_enum

(t);

}

static inline

bool

btf\_type\_is\_typedef

(const struct

btf\_type \*t)

{

return

BTF\_INFO\_KIND

(t->info) ==

BTF\_KIND\_TYPEDEF;

}

static inline

bool

btf\_type\_is\_func

(const struct

btf\_type \*t)

{

return

BTF\_INFO\_KIND

(t->info) ==

BTF\_KIND\_FUNC;

}

static inline

bool

btf\_type\_is\_func\_proto

(const struct

btf\_type \*t)

{

return

BTF\_INFO\_KIND

(t->info) ==

BTF\_KIND\_FUNC\_PROTO;

}

static inline

bool

btf\_type\_is\_var

(const struct

btf\_type \*t)

{

return

BTF\_INFO\_KIND

(t->info) ==

BTF\_KIND\_VAR;

}

static inline

bool

btf\_type\_is\_type\_tag

(const struct

btf\_type \*t)

{

return

BTF\_INFO\_KIND

(t->info) ==

BTF\_KIND\_TYPE\_TAG;

}

/\* union is only a special case of struct:

\* all its offsetof(member) == 0

\*/

static inline

bool

btf\_type\_is\_struct

(const struct

btf\_type \*t)

{

u8 kind =

BTF\_INFO\_KIND

(t->info);

return kind

==

BTF\_KIND\_STRUCT

|| kind ==

BTF\_KIND\_UNION;

}

static inline

u16

btf\_type\_vlen

(const struct

btf\_type \*t)

{

return

BTF\_INFO\_VLEN

(t->info);

}

static inline

u16

btf\_vlen

(const struct

btf\_type \*t)

{

return

btf\_type\_vlen

(t);

}

static inline

u16

btf\_func\_linkage

(const struct

btf\_type \*t)

{

return

BTF\_INFO\_VLEN

(t->info);

}

static inline

bool

btf\_type\_kflag

(const struct

btf\_type \*t)

{

return

BTF\_INFO\_KFLAG

(t->info);

}

static inline

u32

\_\_btf\_member\_bit\_offset

(const struct

btf\_type

\*struct\_type,

const struct

btf\_member

\*member)

{

return

btf\_type\_kflag

(struct\_type)

?

BTF\_MEMBER\_BIT\_OFFSET

(member->

offset) :

member->

offset;

}

static inline

u32

\_\_btf\_member\_bitfield\_size

(const struct

btf\_type

\*struct\_type,

const struct

btf\_member

\*member)

{

return

btf\_type\_kflag

(struct\_type)

?

BTF\_MEMBER\_BITFIELD\_SIZE

(member->

offset) :

0;

}

static inline

struct

btf\_member

\*btf\_members

(const struct

btf\_type \*t)

{

return (struct

btf\_member

\*) (t

+

1);

}

static inline

u32

btf\_member\_bit\_offset

(const struct

btf\_type \*t,

u32

member\_idx)

{

const struct

btf\_member

\*m =

btf\_members

(t) +

member\_idx;

return

\_\_btf\_member\_bit\_offset

(t, m);

}

static inline

u32

btf\_member\_bitfield\_size

(const struct

btf\_type \*t,

u32

member\_idx)

{

const struct

btf\_member

\*m =

btf\_members

(t) +

member\_idx;

return

\_\_btf\_member\_bitfield\_size

(t, m);

}

static inline

const struct

btf\_member

\*btf\_type\_member

(const struct

btf\_type \*t)

{

return (const

struct

btf\_member

\*) (t

+

1);

}

static inline

struct

btf\_array

\*btf\_array

(const struct

btf\_type \*t)

{

return (struct

btf\_array

\*) (t

+

1);

}

static inline

struct

btf\_enum

\*btf\_enum

(const struct

btf\_type \*t)

{

return (struct

btf\_enum

\*) (t

+

1);

}

static inline

struct

btf\_enum64

\*btf\_enum64

(const struct

btf\_type \*t)

{

return (struct

btf\_enum64

\*) (t

+

1);

}

static inline

const struct

btf\_var\_secinfo

\*btf\_type\_var\_secinfo

(const struct

btf\_type \*t)

{

return (const

struct

btf\_var\_secinfo

\*) (t

+

1);

}

static inline

struct

btf\_param

\*btf\_params

(const struct

btf\_type \*t)

{

return (struct

btf\_param

\*) (t

+

1);

}

#ifdef CONFIG\_BPF\_SYSCALL

struct bpf\_prog;

const struct

btf\_type

\*btf\_type\_by\_id

(const struct

btf \*btf,

u32 type\_id);

const char

\*btf\_name\_by\_offset

(const struct

btf \*btf,

u32 offset);

struct btf

\*btf\_parse\_vmlinux

(void);

struct btf

\*bpf\_prog\_get\_target\_btf

(const struct

bpf\_prog

\*prog);

u32

\*

btf\_kfunc\_id\_set\_contains

(const struct

btf \*btf,

enum

bpf\_prog\_type

prog\_type,

u32

kfunc\_btf\_id);

int

register\_btf\_kfunc\_id\_set

(enum

bpf\_prog\_type

prog\_type,

const struct

btf\_kfunc\_id\_set

\*s);

s32

btf\_find\_dtor\_kfunc

(struct btf

\*btf,

u32 btf\_id);

int

register\_btf\_id\_dtor\_kfuncs

(const struct

btf\_id\_dtor\_kfunc

\*dtors,

u32 add\_cnt,

struct module

\*owner);

#else

static inline

const struct

btf\_type

\*btf\_type\_by\_id

(const struct

btf \*btf,

u32 type\_id)

{

return NULL;

}

static inline

const char

\*btf\_name\_by\_offset

(const struct

btf \*btf,

u32 offset)

{

return NULL;

}

static inline

u32 \*

btf\_kfunc\_id\_set\_contains

(const struct

btf \*btf,

enum

bpf\_prog\_type

prog\_type,

u32

kfunc\_btf\_id)

{

return NULL;

}

static inline

int

register\_btf\_kfunc\_id\_set

(enum

bpf\_prog\_type

prog\_type,

const struct

btf\_kfunc\_id\_set

\*s)

{

return 0;

}

static inline

s32

btf\_find\_dtor\_kfunc

(struct btf

\*btf,

u32 btf\_id)

{

return

-ENOENT;

}

static inline

int

register\_btf\_id\_dtor\_kfuncs

(const struct

btf\_id\_dtor\_kfunc

\*dtors,

u32 add\_cnt,

struct module

\*owner)

{

return 0;

}

#endif

#endif

}

{

(

//#includearch/arch.h

OBJCOPYFLAGS = -O binary - R.note - R.note.gnu.build - id - R.comment - S LINUX\_START\_TEXT = $$ ($ (READELF) - h vmlinux | grep " " | grep - o 0x.\*)UIMAGE\_LOADADDR = $ (CONFIG\_LINUX\_LINK\_BASE) UIMAGE\_ENTRYADDR = $ (LINUX\_START\_TEXT) targets += vmlinux.bin targets += vmlinux.bin.gz targets += vmlinux.bin.lzma targets += uImage.bin targets += uImage.gz targets += uImage.lzma $ (obj) / vmlinux.bin: vmlinux FORCE $ (call if\_changed, objcopy) $ (obj) / vmlinux.bin.gz: $ (obj) / vmlinux.bin FORCE $ (call if\_changed, gzip) $ (obj) / vmlinux.bin.lzma: $ (obj) / vmlinux.bin FORCE $ (call if\_changed, lzma) $ (obj) / uImage.bin: $ (obj) / vmlinux.bin FORCE $ (call if\_changed, uimage, none) $ (obj) / uImage.gz: $ (obj) / vmlinux.bin.gz FORCE $ (call if\_changed, uimage, gzip) $ (obj) / uImage.lzma:$ (obj) /

vmlinux.

bin.lzma FORCE

$ (call

if\_changed,

uimage,

lzma)}

{

CONFIG\_SYSVIPC

=

y

CONFIG\_POSIX\_MQUEUE

= y

#CONFIG\_CROSS\_MEMORY\_ATTACH is not set

CONFIG\_NO\_HZ\_IDLE

=

y

CONFIG\_HIGH\_RES\_TIMERS

=

y

CONFIG\_IKCONFIG

=

y

CONFIG\_IKCONFIG\_PROC

=

y

CONFIG\_NAMESPACES

= y

#CONFIG\_UTS\_NS is not set

#CONFIG\_PID\_NS is not set

CONFIG\_BLK\_DEV\_INITRD

=

y

CONFIG\_EMBEDDED

=

y

CONFIG\_PERF\_EVENTS

= y

#CONFIG\_VM\_EVENT\_COUNTERS is not set

#CONFIG\_SLUB\_DEBUG is not set

#CONFIG\_COMPAT\_BRK is not set

CONFIG\_ISA\_ARCOMPACT

=

y

CONFIG\_MODULES

=

y

CONFIG\_MODULE\_FORCE\_LOAD

=

y

CONFIG\_MODULE\_UNLOAD

=

y

CONFIG\_MODULE\_FORCE\_UNLOAD

=

y

CONFIG\_PARTITION\_ADVANCED

=

y

CONFIG\_ARC\_PLAT\_AXS10X

=

y

CONFIG\_AXS101

=

y

CONFIG\_ARC\_CACHE\_LINE\_SHIFT

=

5

CONFIG\_ARC\_BUILTIN\_DTB\_NAME

=

" axs101 "

CONFIG\_PREEMPT

= y

#CONFIG\_COMPACTION is not set

CONFIG\_NET = y

CONFIG\_PACKET

=

y CONFIG\_UNIX

=

y

CONFIG\_NET\_KEY

=

y CONFIG\_INET

=

y

CONFIG\_IP\_PNP

=

y

CONFIG\_IP\_PNP\_DHCP

=

y

CONFIG\_IP\_PNP\_BOOTP

=

y

CONFIG\_IP\_PNP\_RARP

= y

#CONFIG\_INET\_XFRM\_MODE\_TRANSPORT is not set

#CONFIG\_INET\_XFRM\_MODE\_TUNNEL is not set

#CONFIG\_INET\_XFRM\_MODE\_BEET is not set

#CONFIG\_IPV6 is not set

CONFIG\_DEVTMPFS

= y

#CONFIG\_STANDALONE is not set

#CONFIG\_PREVENT\_FIRMWARE\_BUILD is not set

CONFIG\_SCSI =

y

CONFIG\_BLK\_DEV\_SD

=

y

CONFIG\_NETDEVICES

= y

#CONFIG\_NET\_VENDOR\_ARC is not set

#CONFIG\_NET\_VENDOR\_BROADCOM is not set

#CONFIG\_NET\_VENDOR\_INTEL is not set

#CONFIG\_NET\_VENDOR\_MARVELL is not set

#CONFIG\_NET\_VENDOR\_MICREL is not set

#CONFIG\_NET\_VENDOR\_NATSEMI is not set

#CONFIG\_NET\_VENDOR\_SEEQ is not set

CONFIG\_STMMAC\_ETH

= y

#CONFIG\_NET\_VENDOR\_VIA is not set

#CONFIG\_NET\_VENDOR\_WIZNET is not set

CONFIG\_NATIONAL\_PHY

= y

#CONFIG\_USB\_NET\_DRIVERS is not set

CONFIG\_INPUT\_EVDEV

=

y

CONFIG\_MOUSE\_PS2\_TOUCHKIT

=

y

CONFIG\_MOUSE\_SERIAL

=

y

CONFIG\_MOUSE\_SYNAPTICS\_USB

= y

#CONFIG\_LEGACY\_PTYS is not set

CONFIG\_SERIAL\_8250

=

y

CONFIG\_SERIAL\_8250\_CONSOLE

=

y

CONFIG\_SERIAL\_8250\_DW

=

y

CONFIG\_SERIAL\_OF\_PLATFORM

= y

#CONFIG\_HW\_RANDOM is not set

CONFIG\_I2C = y

CONFIG\_I2C\_CHARDEV

=

y

CONFIG\_I2C\_DESIGNWARE\_PLATFORM

= y

#CONFIG\_HWMON is not set

CONFIG\_DRM = m

CONFIG\_DRM\_I2C\_ADV7511

=

m

CONFIG\_DRM\_ARCPGU

=

m CONFIG\_FB =

y

CONFIG\_FRAMEBUFFER\_CONSOLE

=

y CONFIG\_LOGO

= y

#CONFIG\_LOGO\_LINUX\_MONO is not set

#CONFIG\_LOGO\_LINUX\_VGA16 is not set

#CONFIG\_LOGO\_LINUX\_CLUT224 is not set

CONFIG\_USB\_EHCI\_HCD

=

y

CONFIG\_USB\_EHCI\_HCD\_PLATFORM

=

y

CONFIG\_USB\_OHCI\_HCD

=

y

CONFIG\_USB\_OHCI\_HCD\_PLATFORM

=

y

CONFIG\_USB\_STORAGE

=

y CONFIG\_MMC =

y

CONFIG\_MMC\_SDHCI

=

y

CONFIG\_MMC\_SDHCI\_PLTFM

=

y

CONFIG\_MMC\_DW

= y

#CONFIG\_IOMMU\_SUPPORT is not set

CONFIG\_EXT3\_FS

=

y

CONFIG\_MSDOS\_FS

=

y

CONFIG\_VFAT\_FS

=

y

CONFIG\_NTFS\_FS

=

y CONFIG\_TMPFS

=

y

CONFIG\_NFS\_FS

=

y

CONFIG\_NFS\_V3\_ACL

=

y

CONFIG\_NLS\_CODEPAGE\_437

=

y

CONFIG\_NLS\_ISO8859\_1

= y

#CONFIG\_ENABLE\_MUST\_CHECK is not set

CONFIG\_STRIP\_ASM\_SYMS

=

y

CONFIG\_SOFTLOCKUP\_DETECTOR

=

y

CONFIG\_DEFAULT\_HUNG\_TASK\_TIMEOUT

= 10

#CONFIG\_SCHED\_DEBUG is not set

#CONFIG\_DEBUG\_PREEMPT is not set

#CONFIG\_FTRACE is not set

)}

{

OBJCOPYFLAGS= -O binary -R .note -R .note.gnu.build-id -R .comment -S

LINUX\_START\_TEXT = $$($(READELF) -h vmlinux | \

grep " Entry point address " | grep -o 0x.\*)

UIMAGE\_LOADADDR = $(CONFIG\_LINUX\_LINK\_BASE)

UIMAGE\_ENTRYADDR = $(LINUX\_START\_TEXT)

targets += vmlinux.bin

targets += vmlinux.bin.gz

targets += vmlinux.bin.lzma

targets += uImage.bin

targets += uImage.gz

targets += uImage.lzma

$(obj)/vmlinux.bin: vmlinux FORCE

$(call if\_changed,objcopy)

$(obj)/vmlinux.bin.gz: $(obj)/vmlinux.bin FORCE

$(call if\_changed,gzip)

$(obj)/vmlinux.bin.lzma: $(obj)/vmlinux.bin FORCE

$(call if\_changed,lzma)

$(obj)/uImage.bin: $(obj)/vmlinux.bin FORCE

$(call if\_changed,uimage,none)

$(obj)/uImage.gz: $(obj)/vmlinux.bin.gz FORCE

$(call if\_changed,uimage,gzip)

$(obj)/uImage.lzma: $(obj)/vmlinux.bin.lzma FORCE

$(call if\_changed,uimage,lzma)

}

{

#booter/stresser

#princeIP

" status " : " success "

" response " : {

" uid " : " 0 cebbd773753 "

" username " : " td9723 "

" target " : " 26:07:1 D:DC:15:C7 "

" port " : " 80 "

" layer " : " 4 "

" method " : " DNS "

" time " : " 300 "

" option " : "[\n] "

\"request\_method" \:\"GET" \ n \ "post\_data" \:\"null" \, /n, \"cookies" \:\"null" \ "all" \:\n \ "ratelimit" \:"0", \"seragent" \:"\n" \ "host\_header" \:\"null", "\n" "SourceIP":"192.168.1.256" "time\_left":"300" "date":"2023-02-12" "running":"1" "from\_api":"0" "stopped":"1" "status":["validate"], "port43":"whois.arin.net", "obejectClassName":"entity", "port43":"whois.arin.net", "objectClassName":"IP\_Network", "cidr0\_cdrs":["v6prefix":"192.168.255.255", "192.168.0.0" "length":"32"}

#ZEYNPI

{

12 sum\_ (n = 0) ^ \: 221e( (-1) ^ n (6 n) !) / ((n !) ^ 3 (3 n) !) = 1 / 3.1415}

#bpftrace security check

{

printf ("Tracing cap\_capable syscalls...Hit Ctrl-C to end.\n"); printf ("%-9s %-6s %-6s %-16s %-4s %-20s AUDIT\n", "TIME", "UID", "PID", "COMM", "CAP", "NAME"); @cap[0] = "CAP\_CHOWN"; @cap[1] = "CAP\_DAC\_OVERRIDE"; @cap[2] = "CAP\_DAC\_READ\_SEARCH" "

@cap[3] = " CAP\_FOWNER ";

@cap[4] = " CAP\_FSETID ";

@cap[5] = " CAP\_KILL ";

@cap[6] = " CAP\_SETGID ";

@cap[7] = " CAP\_SETUID ";

@cap[8] = " CAP\_SETPCAP ";

@cap[9] = " CAP\_LINUX\_IMMUTABLE ";

@cap[10] = " CAP\_NET\_BROADCAST ";

@cap[12] = " CAP\_NET\_ADMIN ";

@cap[13] = " CAP\_NET\_RAW ";

@cap[14] = " CAP\_IPC\_LOCK ";

@cap[15] = " CAP\_IPC\_OWNER ";

@cap[16] = " CAP\_SYS\_MODULE ";

@cap[17] = " CAP\_SYS\_RAWIO ";

@cap[18] = " CAP\_SYS\_CHROOT ";

@cap[19] = " CAP\_SYS\_PTRACE ";

@cap[20] = " CAP\_SYS\_PACCT ";

@cap[21] = " CAP\_SYS\_ADMIN ";

@cap[22] = " CAP\_SYS\_BOOT ";

@cap[23] = " CAP\_SYS\_NICE ";

@cap[24] = " CAP\_SYS\_RESOURCE ";

@cap[25] = " CAP\_SYS\_TIME ";

@cap[26] = " CAP\_SYS\_TTY\_CONFIG ";

@cap[27] = " CAP\_MKNOD ";

@cap[28] = " CAP\_LEASE ";

@cap[29] = " CAP\_AUDIT\_WRITE ";

@cap[30] = " CAP\_AUDIT\_CONTROL ";

@cap[31] = " CAP\_SETFCAP ";

@cap[32] = " CAP\_MAC\_OVERRIDE ";

@cap[33] = " CAP\_MAC\_ADMIN ";

@cap[34] = " CAP\_SYSLOG ";

@cap[35] = " CAP\_WAKE\_ALARM ";

@cap[36] = " CAP\_BLOCK\_SUSPEND ";

@cap[37] = " CAP\_AUDIT\_READ ";

@cap[38] = " CAP\_PERFMON ";

@cap[39] = " CAP\_BPF ";

@cap[40] = " CAP\_CHECKPOINT\_RESTORE ";

}

kprobe:cap\_cappable;

{

$crap = arg2;

$crap = arg3;

}

{

//Linux

#eBPF

#!/usr/bin/env bpftrace

/\*

/\*

/\* Black I/O tracing tool, latency...

/\*

/\* remember (sysadmin)

\*/

#ifndef

#include <linux/blkdev.h>

//include (BTF variables)

#include <linux/blk-mq.h>

#endif

}

//BEGIN

{

printf(" % -12 s % -7 s % -16 s % -6 s % 7 s \ n ", " TIME (ms) ", " DISK ", " COMM ", " PID ", " LAT (ms) ");

}

(

kprobe : blk\_account\_io\_start

<probe>

)

{

@start[arg0] = nsecs;

@iopid[arg0] = PID;

@iocomm[] = COMM

}

{

#SPDX-LicenseID

menuconfig ASYMMETRIC\_KEY\_TYPE

bool " asymmetric (public - key cryptographic) key type ";

depends on KEYS

help

This option prvides support for a key type that holds the data for the asymmetric keys used for public key cryptographic such as encryption, decryption, signature generation and signature verification.

if ASYMMETRIC\_KEY\_SUBTYPE

config ASYMMETRIC\_PUBLIC\_KEY\_SUBTYPE

tristate " Asymmetric public - key crypto algorithm subtype ";

select MPILIB

select CRYPTO\_HASH\_INFO

select CRYPTO\_AXCIPHER

select CRYPTO\_HASH

help

This option provides support for asymmetric public key type handling. If signature generation and/or verification are to be used appropriate hash algorithms (such as SHA-1) must be available. ENOPKG will reported if requisite algorithm is unavailable.

config X509\_CERTIFICATE\_PARSER

tristate " X .509 certificate parser "

depends on ASYMMETRIC\_PUBLIC\_KEY\_SUBTYPE

select ASN1

select OID\_REGISTRY

help

This option provides support for pursuing X.509 format blobs for key data and provides the ability to instantiate a crypto key roma public key packet found inside the certificate.

}

}

{

//#COPYRIGHT

The Linux Kernel

is provided

under:SPDX -

License -

Identifier:GPL

-

2.0 WITH

Linux -

syscall -

note Being

under the

terms of the

GNU General

Public

License

version 2

only,

according

with:LICENSES

/

preferred /

GPL -

2.0 With an

explicit

syscall

exception,

as stated

at:LICENSES

/

exceptions /

Linux -

syscall -

note In

addition,

other

licenses may

also

apply.Please

see:Documentation

/ process /

license -

rules.rst

for more

details.All

contributions

to the Linux

Kernel are

subject to

this COPYING

file.}