PUT ON YOUR TINFO_T HAT

MIAUBIZ

DEFCON 23

AUGUST 7TH 2015


TEXAS
MTAUBTZ
ESTD 1845


TEXAS
AZIMUTH
ESTD 1845

C++ templates

```
template <typename Type>
Type max(Type a, Type b) {
    return a > b ? a : b;
}
```

specialized

```
char max<char>(char a, char b) {
    return a > b ? a : b;
}
```

specialized

```
uint32_t max<uint32_t>(uint32_t a, uint32_t b)
{

    return a > b ? a : b;

}
```

C++ Templates

- `anonymous namespace'::OpenBSDTargetInfo<`anonymous namespace'::SparcV8TargetInfo>::getOSDefines(clang::LangOptions const&,llvm::Triple const&,clang::MacroBuilder &)

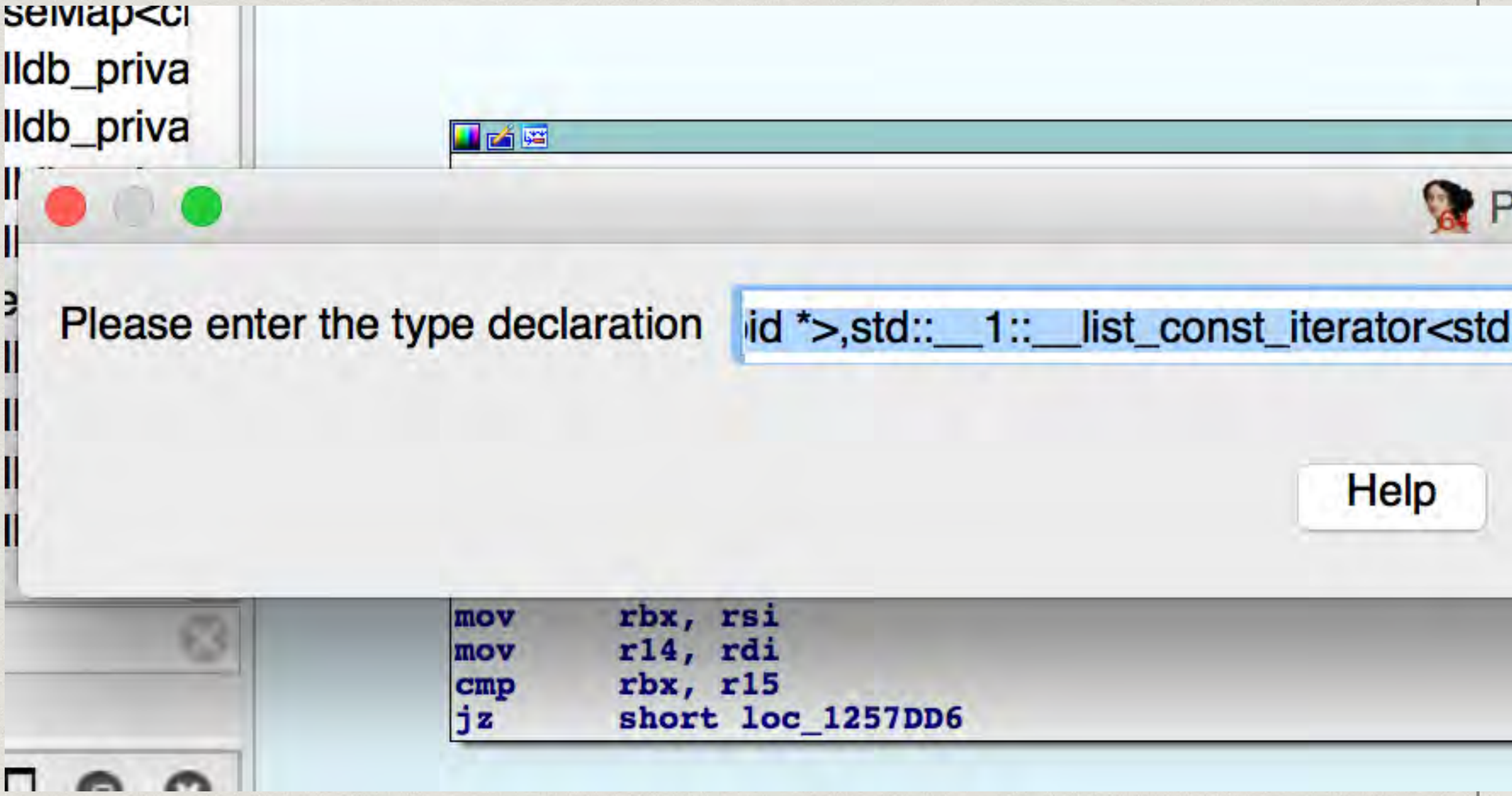
templates

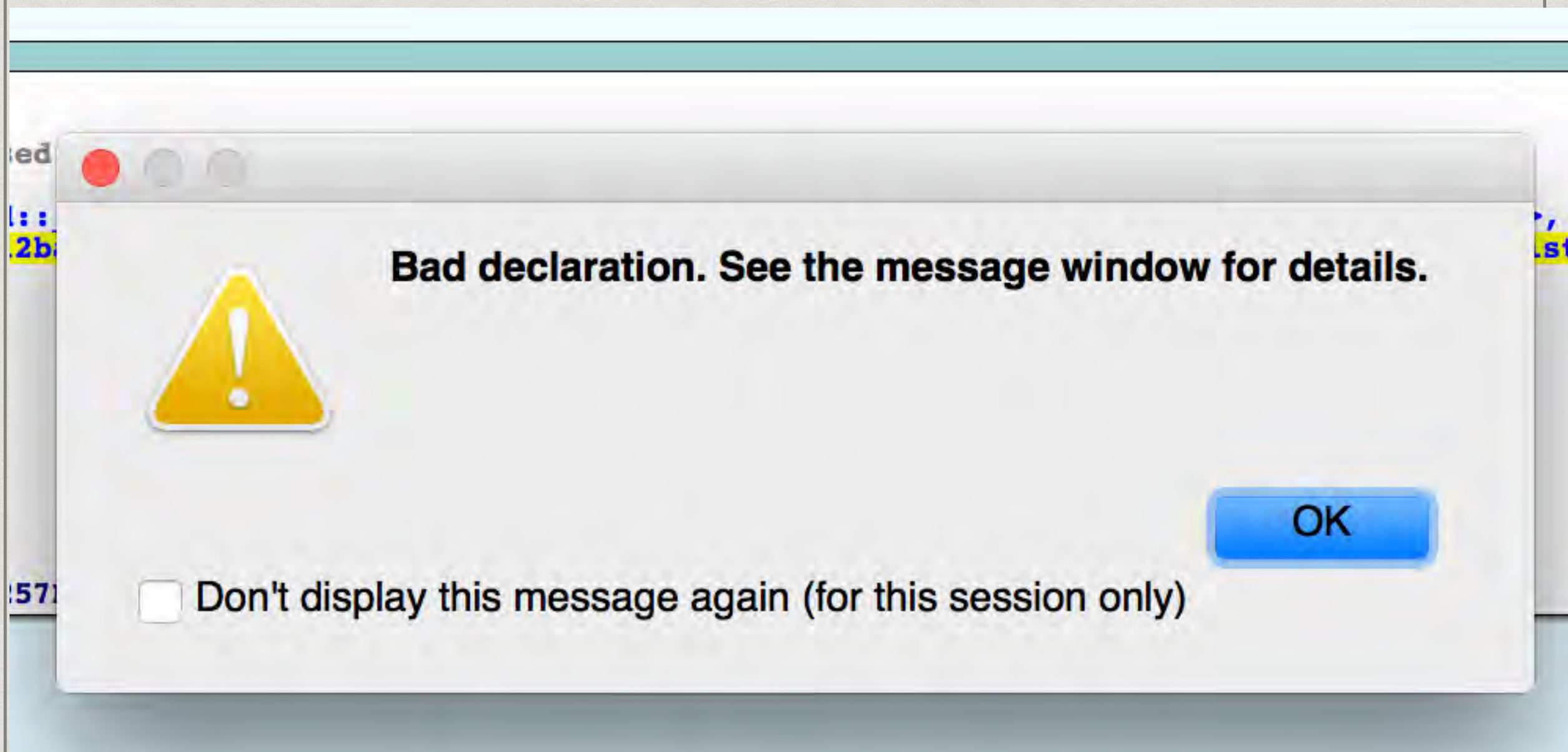
- std::__1::list<std::__1::basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>>,std::__1::allocator<std::__1::basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>>>>::erase(std::__1::__list_const_iterator<std::__1::basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>>,void *>,std::__1::__list_const_iterator<std::__1::basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>>,void *>)


TEXAS
1DAPRO
OIL AND GAS

```
; Attributes: bp-based frame

; std::__1::list<std::__1::basic_string<char, std::__1
__ZNSt3__14listINS_12basic_stringIcNS_11char_traitsIcE
push    rbp
mov     rbp, rsp
push    r15
push    r14
push    r12
push    rbx
mov     r15, rdx
mov     rbx, rsi
mov     r14, rdi
cmp     rbx, r15
jz      short loc_1257DD6
```


lldb_priva
lldb_priva
Please enter the type declaration
id *>,std::__1::__list_const_iterator<std
Help
mov rbx, rsi
mov r14, rdi
cmp rbx, r15
jz short loc_1257DD6


Bad declaration. See the message window for details.
OK
Don't display this message again (for this session only)

Output window

```
Syntax error near: std::__1::list
Command "SetType" failed
```

THREE WAYS TO FIX THIS.


TEXAS
PYTHON
DON'T TREAD ON ME

SOLUTION 1

idc.GetTinfo

- `Python> idc.GetTinfo(idc.here())`
- `('\x0cp\x05\x03\xffA\n=\x04#\x99m\n=\x04#\x95v', '\x05this\x01')`

idc.GetTinfo

- the type: “\x0cp\x05\x03\xffA \n=\x04#\x99m\n=\x04#\x95v”
- the args: “\x05this\x01”

idc.ApplyType

- `idc.ApplyType(ea, idc.GetTinfo(ea))`
- `True`

reverse the IDA

- \x0cp\x05\x03\xffA\n=\x04#\x99m\n=\x04#\x95v

the juicy bit

\n=	\x04	#	\x99m
\n=	\x04	#	\x95v
* etc	length	ordinal follows	the ordinal

Local Type Ordinals

Local Types | IDA View-A | Pseudoc

Ordinal	Name
1231	CommandObjectTargetModulesSearchPathsClea
1232	CommandObjectTargetModulesSearchPathsAdc
1233	CommandObjectTargetModulesLookup
1234	CommandObjectTargetModulesLookup::Comma
1235	CommandObjectTargetModulesList
1236	CommandObjectTargetModulesList::CommandC
1237	CommandObjectTargetModulesDump
1238	CommandObjectTargetModulesDumpLineTable
1239	CommandObjectTargetModulesSourceFileAutoC
1240	CommandObjectTargetModulesDumpSymfile
1241	CommandObjectTargetModulesModuleAutoCom
1242	CommandObjectTargetModulesDumpSections

get ordinal by name

```
def find_type_by_name(self, name):
  for i in range(1, GetMaxLocalType()):
    if name == GetLocalTypeName(i):
      return i
  print "didn't find: %s" % name
  raise RuntimeError
```

decode the ordinal

```
\x99m -> 0x996d
```

decode the ordinal

```
0x996d ->
(0x99&0x7f) * 0x40 +
  (0x6d & ~0x40)
-> 1645
```

decode the ordinal

```
\x95v -> 0x9575
```

decode the ordinal

```
0x9575 ->
(0x95 & 0x7f) * 0x40 +
(0x76 & ~0x40)

-> 1398
```

so 1645 and 1398

Hex View-1 | Structures | Enums | Imports | Expor

```
ctTypeFormatAdd::CommandOptions *this, lldb_private::CommandInterpreter *a2)
```

Ordinal	Name
1645	CommandObjectTypeFormatAdd::CommandOptions

Unexplored | Instruction | External sy

Local Types | IDA View-A

Ordinal	Name
1398	lldb_private::CommandInterpreter

encode an ordinal

```
((ordinal >> 6) & 0xff | 0x80) +
(ordinal & 0x3f | 0x40)
```

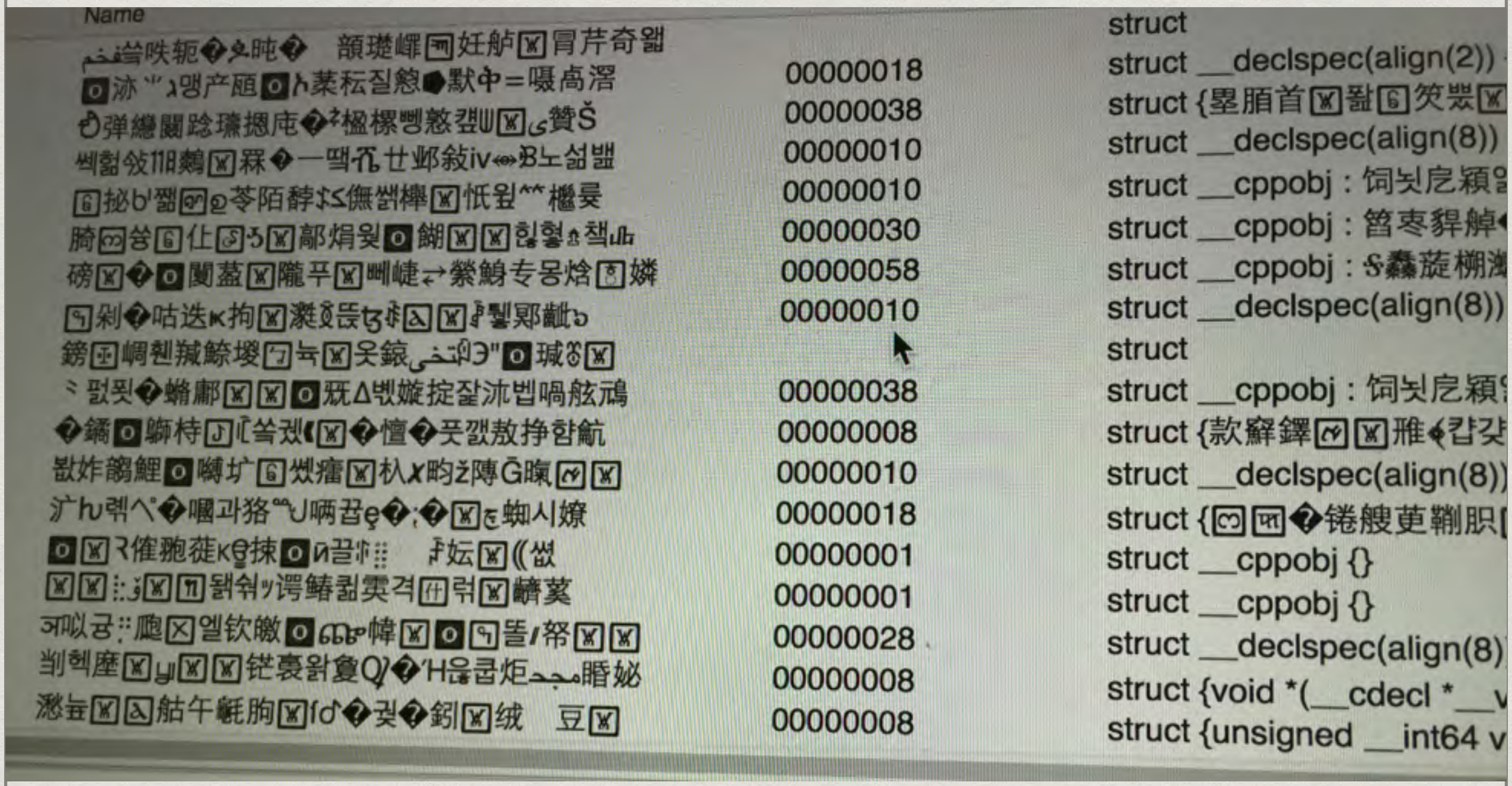
reconstitute the string

```
Python>
idc.ApplyType(idc.here(),
('\x0cp\x05\x03\xffA\n=\x04#AA
\n=\x04#bb', '\x05this\x01'))

True
```

SOLUTION 2:
TYPES CAN HAVE ANY NAME

types can have any name



rename all types

- rename all involved types to benign names
- set types
- rename types back

this API does what it says

```
idaapi.rename_named_type(
idaapi.cvar.idati,
CURRENT_NAME, NEW_NAME,
idaapi.NTF_TYPE);
```

SOLUTION 3:
TINFO_T APIS


TEXAS
HEXRAYS
NASCAR

tinfo_t yeah

```
vdui = get_tform_vdui(get_current_tform())
de = idaapi.decompile(idc.here())
lvar = de.arguments[0]
a_type = idaapi.tinfo_t()
a_type.get_named_type(idaapi.cvar.idati,
“some<crazy, name<lol>>”);
ptr_type = tinfo_t()
ptr_type.create_ptr(a_type)
vdui.set_lvar_type(lvar, ptr_type)
```

tinfo_t yeah

```
a_type = idaapi.tinfo_t()
a_type.get_named_type(
idaapi.cvar.idati,
“some<crazy, name<lol>>”)
```

tinfo_t yeah

```
vdui.set_lvar_type(
  lvar, a_type)
```

and the return type?

- there is no lvar that changes the return type of a function
- can’t add / remove arguments

SOLUTION 3.1:

CREATE AN ARBITRARY FUNCTION TYPE

create a function type

```
_, tp, _ = idc.ParseType(
“static volatile OWORD*
__fastcall lol(static volatile
unsigned double);”, PT_TYP)
```

use weird marker types no real person would use.
put in the number of arguments you want.

create a function type

```
tp_replaced =
do_stuff_from_solution1(tp)
```

create a function type

```
fn_tinfo = tinfo_t()
fn_tinfo.deserialize(cvar.idati,
                     tp_replaced, “”)
```

create a function type

```
apply_tinfo2(ea,
fn_tinfo, TINFO_DEFINITE
```

(apply_tinfo doesn't apply tinfo)

create a function type

```
_, tp, _ = idc.ParseType(“static volatile
OWORD *__fastcall lol(static volatile
unsigned double);”, PT_TYP)
tp_replaced = do_stuff_from_solution1(tp)
fn_tinfo = tinfo_t()
fn_tinfo.deserialize(cvar.idati,
tp_replaced, “”)
apply_tinfo2(ea, fn_tinfo, TINFO_DEFINITE)
```


TEXAS
DEFCON
Rangorettes
Kilgore College