Subject: Bug (?) in ide\Debuggers\Exp.cpp
Posted by mr_ped on Tue, 23 May 2006 10:25:56 GMT
View Forum Message <> Reply to Message

(I'm just trying to figure out why the "data[2]" does lead to "Only pointer can be dereferenced" ... still reading the Exp.cpp + parser.cpp classes, I will need probably some more time (like couple of days, as I have just couple of minutes per day for Ultimate++) to get familiar with those classes)

I found out suspicious code in Exp.cpp (line 136, 605dev1 src):

```
Pdb::Val Pdb::Compute(Pdb::Val v1, Pdb::Val v2, int oper)
{
 if(v1.ref) {
  int q = (int)GetInt(v2) * (v1.ref > 1 ? 4 : SizeOfType(v1.type));
  v1 = GetRVal(v1);
  switch(oper) {
  case '+': v1.address += q; break;
  case '-': v1.address -= q; break;
  default: ThrowError("Invalid pointer arithmetics");
  }
  return v1;
 }
 if(v1.ref) {
  int q = (int)GetInt(v1) * (v2.ref ? 4 : SizeOfType(v2.type));
  v2 = GetRVal(v2);
  if(oper == '+')
   v2.address += q;
  else
   ThrowError("Invalid pointer arithmetics");
  return v2;
 }
```

Looks to me like the second if should be:
if (v2.ref) {

Besides that it looks to me like (almost) duplicite code... which I personally quite dislike, when it can be avoided easily.
Maybe some

```
Pdb::val & _v1 = v1;
Pdb::val & _v2 = v2;
if (!v1.ref && v2.ref) _v1 = v2, _v2 = v1;
//further _v1 and _v2 are used instead of v1/v2
```

would be more elegant and more powerful (supporting "-" operator even if v2 is "ref"), but I don't have enough insight into the code to judge whether the "more powerful" part is actually needed, or counterproductive. And it would maybe still look somewhat cumbersome anyway. (And I didn't try

that proposal, so I'm not sure if it doesn't have some further catch.)

Edit: now I'm not sure if _v1 = v2 would not overwrite also original v1 content ... probably yes?
Once the reference is set, how to change it in C++?
I already forgot how references work exactly (as I do use plain C in work right now, for last year or so)... Maybe I should check the C++ language reference again.

Subject: Re: Bug (?) in ide\Debuggers\Exp.cpp
Posted by mirek on Tue, 23 May 2006 10:41:13 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Tue, 23 May 2006 06:25(I'm just trying to figure out why the "data[2]" does lead to "Only pointer can be dereferenced" ... still reading the Exp.cpp + parser.cpp classes, I will need probably some more time (like couple of days, as I have just couple of minutes per day for Ultimate++) to get familiar with those classes)

I found out suspicious code in Exp.cpp (line 136, 605dev1 src):

```
Pdb::Val Pdb::Compute(Pdb::Val v1, Pdb::Val v2, int oper)
{
 if(v1.ref) {
  int q = (int)GetInt(v2) * (v1.ref > 1 ? 4 : SizeOfType(v1.type));
  v1 = GetRVal(v1);
  switch(oper) {
  case '+': v1.address += q; break;
  case '-': v1.address -= q; break;
  default: ThrowError("Invalid pointer arithmetics");
  }
  return v1;
 }
 if(v1.ref) {
  int q = (int)GetInt(v1) * (v2.ref ? 4 : SizeOfType(v2.type));
  v2 = GetRVal(v2);
  if(oper == '+')
   v2.address += q;
  else
   ThrowError("Invalid pointer arithmetics");
  return v2;
 }
```

Looks to me like the second if should be:
if (v2.ref) {

Besides that it looks to me like (almost) duplicite code... which I personally quite dislike, when it can be avoided easily.
Maybe some

```
Pdb::val & _v1 = v1;
Pdb::val & _v2 = v2;
if (!v1.ref && v2.ref) _v1 = v2, _v2 = v1;
//further _v1 and _v2 are used instead of v1/v2
```

would be more elegant and more powerful (supporting "-" operator even if v2 is "ref"), but I don't have enough insight into the code to judge whether the "more powerful" part is actually needed, or counterproductive. And it would maybe still look somewhat cumbersome anyway. (And I didn't try that proposal, so I'm not sure if it doesn't have some further catch.)

Edit: now I'm not sure if _v1 = v2 would not overwrite also original v1 content ... probably yes?
Once the reference is set, how to change it in C++?
I already forgot how references work exactly (as I do use plain C in work right now, for last year or so)... Maybe I should check the C++ language reference again.

Thanks!

Mirek

Subject: Re: Bug (?) in ide\Debuggers\Exp.cpp
Posted by mr_ped on Tue, 23 May 2006 12:00:45 GMT
View Forum Message <> Reply to Message

And changing the code to:

```
Pdb::Val Pdb::Compute(Pdb::Val v1, Pdb::Val v2, int oper)
{
 if(v1.ref || v1.array) {

...

 if(v2.ref || v2.array) {

...
```

does help in the debugger to allow me to use in Watches things like
data[2] upon data array.
Also works for arrays with elements larger than single byte, everything as expected.

@Mirek: any idea why the above mentioned modification od TheIDE may break something?
I've been thinking about it for a while, and I don't see any potentional harm by changing Pdb::Compute to work with arrays like with pointers...
IMHO working with them in any other way has no purpose, but you are the one to judge this.

If the change makes sense, add it to next dev realease.

Also I think some similar code is sitting somewhere in Assist, which is inhibiting it to work correctly with constructions like

Vector<byte> array_of_vectors[20];

(after array_of_vectors[0]. the Assist is lost and does not offer Vector<T> methods)

Subject: Re: Bug (?) in ide\Debuggers\Exp.cpp
Posted by mr_ped on Tue, 23 May 2006 17:19:45 GMT
View Forum Message <> Reply to Message

I did check Assist a bit... I'm still confused about how exactly the evaluation of type works with all those strings and things like "operator()" in results (running it couple of times in debugger would certainly help me, but I don't know how to debug Assist), but I see it's completely different from pdb approach and definitely not as easy to fix, as I dreamed it will be. Actually quite difficult for newcomer like me.

I will probably give up with the Assist, and try to add some more improvements to debugger, if I will have some spare time.
I think that will be easier for me, the debugger code is pretty clean and easy to work with.

Subject: Re: Bug (?) in ide\Debuggers\Exp.cpp
Posted by mr_ped on Thu, 01 Feb 2007 17:51:20 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Tue, 23 May 2006 14:00And changing the code to:

```
Pdb::Val Pdb::Compute(Pdb::Val v1, Pdb::Val v2, int oper)
{
 if(v1.ref || v1.array) {

...

 if(v2.ref || v2.array) {

...
```

does help in the debugger to allow me to use in Watches things like
data[2] upon data array.
Also works for arrays with elements larger than single byte, everything as expected.

@Mirek: any idea why the above mentioned modification od TheIDE may break something?

I've been thinking about it for a while, and I don't see any potentional harm by changing Pdb::Compute to work with arrays like with pointers...
IMHO working with them in any other way has no purpose, but you are the one to judge this.

If the change makes sense, add it to next dev realease.

Also I think some similar code is sitting somewhere in Assist, which is inhibiting it to work correctly with constructions like

Vector<byte> array_of_vectors[20];

(after array_of_vectors[0]. the Assist is lost and does not offer Vector<T> methods)

@Mirek: did this one make it into UPP?
I didn't check UPP for looooooong time, have been busy with other projects, but right now I'm searching for a C++ IDE for linux for commercial development, so I think I will watch UPP more closely again.
I hope you are glad I'm back. hehe (because I'm)

The last time I checked UPP (can't even remember the version number, must have been a year or so) this fix of array debugging was not added. If not, why not?

Subject: Re: Bug (?) in ide\Debuggers\Exp.cpp
Posted by mirek on Thu, 01 Feb 2007 18:40:57 GMT
View Forum Message <> Reply to Message

Welcome back.

Sorry for not replying about that patch. The reason was that I am not sure what problem it is patching - array access (like in data[2]) is in Pdb::Post.

Can you post a testcase where current code does not work as it should?

Mirek
