
Subject: FDS Crash

Posted by [Neijwiert](#) on Mon, 18 Nov 2013 22:12:43 GMT

[View Forum Message](#) <> [Reply to Message](#)

Order of Crash:

- Join game
- Purchase refill
- Crash

Crash Type: Access violation

Crash points to: Scripts.dll

I know that it must have been my code, but I really can't find it. This is the code that I have that should get executed on refill:

```
// Not sure if it executes OnObjectCreate, but just in case
void NTC::OnObjectCreate(void *data, GameObject *obj)
{
    Console_Output("A");
    NTCPC->OnObjectCreate(obj);
}
```

```
bool NTC::OnRefill(GameObject *purchaser)
{
    Console_Output("Refill\n");
    return true;
}
```

It doesn't hit either of the Console_Output lines.

I'm out of ideas, can you please help me? I'm willing to hand over the complete source code I use.

I tried:

- Debug Text
- Attaching debugger
- Looking in the crashdump

File Attachments

1) [crashdump.20131118-220739-r5704-n1.dmp](#), downloaded 63 times

Subject: Re: FDS Crash

Posted by [Neijwiert](#) on Mon, 18 Nov 2013 22:32:02 GMT

After some more digging I found out that the problem lays at this method:

```
void NTCPlayerControl::Remove_Excess_Scripts(GameObject *Obj)
{
    if(!Obj)
        return;

    // Loop trough all the scripts
    SimpleDynVecClass<GameObjObserverClass *> Observers = Obj->Get_Observers();
    int Count = Observers.Count();
    for(int x = 0; x < Count; x++)
    {
        GameObjObserverClass *Current = Observers[x];
        if(Current)
        {
            WideStringClass Name = Current->Get_Name();

            if(Name.Compare(L"M00_GrantPowerup_Created") == 0 || Name.Substring(0,
4).Compare_No_Case(L"SSGM") == 0)
            {
                COutput("Should remove: %s\n", Current->Get_Name());
                //Obj->Remove_Observer(Current);
            }
        }
    }
}
```

As soon as I add the Obj->Remove_Observer line it starts crashing again. But I still haven't figured out why I cant add that line.

UPDATE:

I think somewhere in Scripts.dll there is code that assumes that all SSGM scripts are still attached.

I've changed the code to this to make sure the comparing is going right:

```
if(Current)
{
    const char *ScriptName = Current->Get_Name();

    //Name.Substring(0, 4).Compare_No_Case(L"SSGM") == 0
    if(strcmp(ScriptName, "M00_GrantPowerup_Created") == 0 || (ScriptName &&
strlen(ScriptName) >= 4 && ScriptName[0] == 'S' && ScriptName[1] == 'S' && ScriptName[2] ==
'G' && ScriptName[3] == 'M'))
    {
        COutput("Should remove: %s\n", Current->Get_Name());
        Obj->Remove_Observer(Current);
    }
}
```

```
}  
}
```

Subject: Re: FDS Crash

Posted by [Neijwiert](#) on Mon, 18 Nov 2013 22:52:38 GMT

[View Forum Message](#) <> [Reply to Message](#)

As soon as I wrote this part "I think somewhere in Scripts.dll there is code that assumes that all SSGM scripts are still attached." and ate a cookie. I realised that it must have been in the SSGM code.

The one who is responsible of SSGM should change these 2 functions:

Toggle Spoiler

```
bool SSGMGameManager::RefillHook(GameObject *purchaser)  
{  
    SSGM_Soldier *script = (SSGM_Soldier *)Find_Script_On_Object(purchaser,"SSGM_Soldier");  
    if (RefillLimit)  
    {  
        if (The_Game()->Get_Game_Duration_S() - script->RefillTime < RefillLimit)  
        {  
            return false;  
        }  
    }  
    script->RefillTime = The_Game()->Get_Game_Duration_S();  
    for (int i = 0;i < RegisteredEvents[EVENT_REFILL_HOOK].Count();i++)  
    {  
        if (!RegisteredEvents[EVENT_REFILL_HOOK][i]->OnRefill(purchaser))  
        {  
            return false;  
        }  
    }  
    return true;  
}
```

```
int SSGMGameManager::CharacterPurchaseHook(BaseControllerClass *base,GameObject  
*purchaser,unsigned int cost,unsigned int preset,const char *data)  
{  
    SSGM_Soldier *script = (SSGM_Soldier *)Find_Script_On_Object(purchaser,"SSGM_Soldier");  
    if (RefillLimit)  
    {  
        if (The_Game()->Get_Game_Duration_S() - script->RefillTime < RefillLimit)  
        {  
            return 4;  
        }  
    }  
}
```

```

if (IsPresetDisabled(preset))
{
    return 4;
}
StringClass str;
const char *str2 = Get_Translated_Definition_Name_Ini(Get_Definition_Name(preset));
const char *str3 = Get_Player_Name(purchaser);
str.Format("Purchase: %s - %s",str3,str2);
delete[] str3;
delete[] str2;
SSGMGameLog::Log_Message(str,"_PURCHASE");
bool fp = FreePurchases;
for (int i = 0;i < RegisteredEvents[EVENT_CHARACTER_PURCHASE_HOOK].Count();i++)
{
    int ret =
RegisteredEvents[EVENT_CHARACTER_PURCHASE_HOOK][i]->OnCharacterPurchase(base,p
urchaser,cost,preset,data);
    if (ret == -2)
    {
        fp = true;
    }
    else if (ret != -1)
    {
        return ret;
    }
}
if (fp)
{
    return -2;
}
return -1;
}

```

to:

Toggle Spoiler

```

bool SSGMGameManager::RefillHook(GameObject *purchaser)
{
    SSGM_Soldier *script = (SSGM_Soldier *)Find_Script_On_Object(purchaser,"SSGM_Soldier");
    if(script)
    {
        if (RefillLimit)
        {
            if (The_Game()->Get_Game_Duration_S() - script->RefillTime < RefillLimit)
            {

```

```

    return false;
}
}
script->RefillTime = The_Game()->Get_Game_Duration_S();
for (int i = 0; i < RegisteredEvents[EVENT_REFILL_HOOK].Count(); i++)
{
    if (!RegisteredEvents[EVENT_REFILL_HOOK][i]->OnRefill(purchaser))
    {
        return false;
    }
}
return true;
}

int SSGMGameManager::CharacterPurchaseHook(BaseControllerClass *base, GameObject
*purchaser, unsigned int cost, unsigned int preset, const char *data)
{
    SSGM_Soldier *script = (SSGM_Soldier *)Find_Script_On_Object(purchaser, "SSGM_Soldier");
    if (script)
    {
        if (RefillLimit)
        {
            if (The_Game()->Get_Game_Duration_S() - script->RefillTime < RefillLimit)
            {
                return 4;
            }
        }
        if (IsPresetDisabled(preset))
        {
            return 4;
        }
        StringClass str;
        const char *str2 = Get_Translated_Definition_Name_Ini(Get_Definition_Name(preset));
        const char *str3 = Get_Player_Name(purchaser);
        str.Format("Purchase: %s - %s", str3, str2);
        delete[] str3;
        delete[] str2;
        SSGMGameLog::Log_Message(str, "_PURCHASE");
        bool fp = FreePurchases;
        for (int i = 0; i < RegisteredEvents[EVENT_CHARACTER_PURCHASE_HOOK].Count(); i++)
        {
            int ret =
RegisteredEvents[EVENT_CHARACTER_PURCHASE_HOOK][i]->OnCharacterPurchase(base, p
urchaser, cost, preset, data);
            if (ret == -2)
            {
                fp = true;
            }
        }
    }
}

```

```
}  
else if (ret != -1)  
{  
    return ret;  
}  
}  
if (fp)  
{  
    return -2;  
}  
}  
return -1;  
}
```

Subject: Re: FDS Crash

Posted by [danpaul88](#) on Tue, 19 Nov 2013 12:50:36 GMT

[View Forum Message](#) <> [Reply to Message](#)

That change would also stop your own custom refill scripts being called since you embedded the event hook loop inside the if(script) block.

Subject: Re: FDS Crash

Posted by [Troll King](#) on Tue, 19 Nov 2013 13:36:39 GMT

[View Forum Message](#) <> [Reply to Message](#)

Oh right, hehe yeah sorry I wasn't really looking at it very well. I guess I was tired. But I think you get the suggestion that there needs to be a null pointer check.

Subject: Re: FDS Crash

Posted by [danpaul88](#) on Tue, 19 Nov 2013 17:43:23 GMT

[View Forum Message](#) <> [Reply to Message](#)

Revision: 6096

Author: danpaul88

Date: 19 November 2013 17:42:18

Message:

Added some null pointers check in the SSGM refill logic to guard against the SSGM_Soldier script somehow being removed from a soldier (ie: by third party code trying to override SSGM functionality with its own)

Modified : /trunk/scripts/scripts/gmgame.cpp
