

Hackerman's Hacking Tutorials

The knowledge of anything, since all things have causes, is not acquired or complete unless it is known by its causes. - Avicenna

[About Me!](#)[Cheat Sheet](#)[My Clone](#)[How This Website is Built](#)[The Other Guy from Wham!](#)

JAN 29, 2019 - 4 MINUTE READ - [COMMENTS](#) - **GAME HACKING**

Cheating at Moonlighter - Part 3 - Enabling Debug HUD

- [Enabling Debug Mode](#)
 - [Debug Shortcuts](#)
 - [KeyCodes](#)
 - [Controller Shortcuts](#)
 - [Save/Reset Shortcuts](#)
 - [Enabling Debug HUD](#)
- [Lessons Learned](#)

In this part, I am going to use dnSpy to enable the Debug HUD. We will analyze how it's enabled and how it can be accessed.

Who am I?

I am Parsia, a security engineer at [Electronic Arts](#).

I write about application security, reverse engineering, Go, cryptography, and (obviously) videogames.

Click on [About Me!](#) to know more.



in

Collections

- [Cheating at Moonlighter - Part 1 - Save File](#)
- [Cheating at Moonlighter - Part 2 - Changing Game Logic with dnSpy](#)

Looking at `StatsModifier.intelligence`, I went down this rabbit hole to see how items are created.

[Thick Client Proxying](#)

[Go/Golang](#)

[Blockchain/Distributed Ledgers](#)

[Automation](#)

[Reverse Engineering](#)

[Crypto\(graphy\)](#)

[CTFs/Writeups](#)

[WinAppDbg](#)

[AWSome.pw - S3 bucket squatting - my very legit branded vulnerability](#)

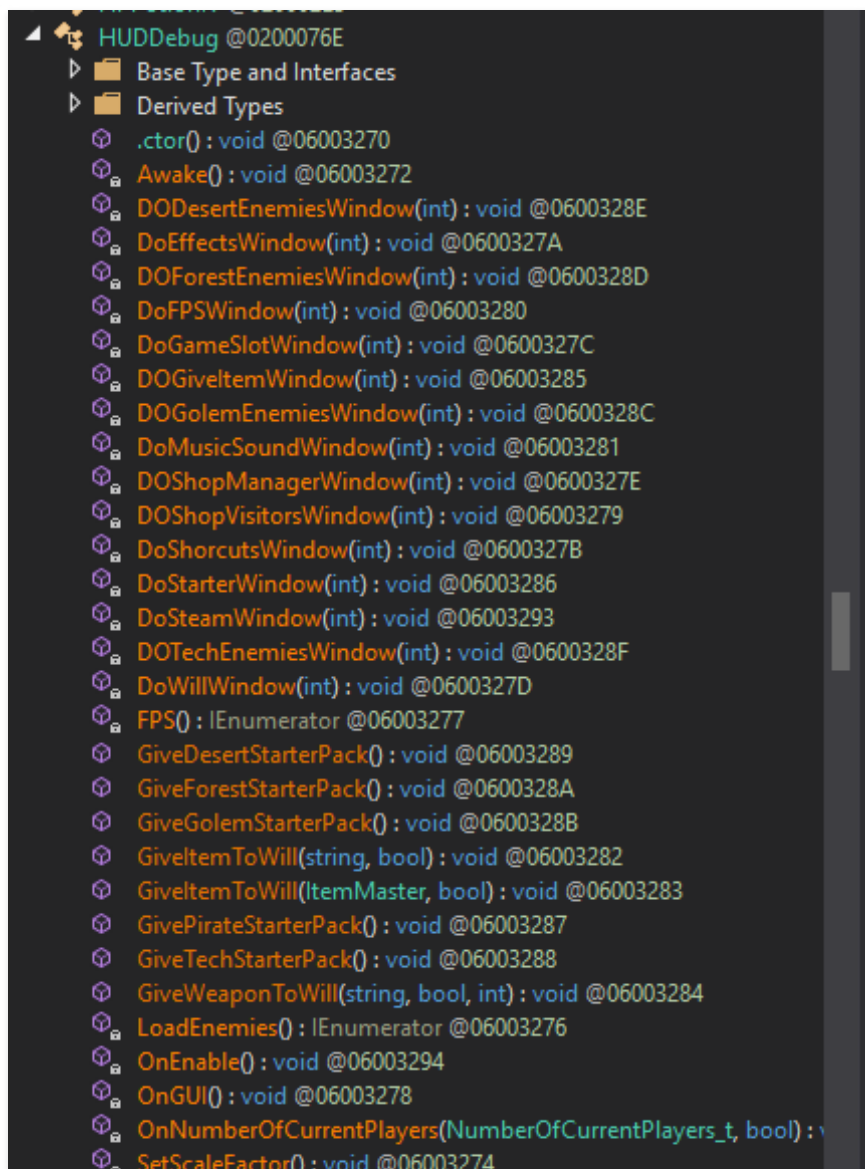
```

StatsModifier.intelligence : int @0400094C
└─ Assigned By
  └─ Read By
    └─ EquipmentStats.AddModifier(StatsModifier) : void @06000F57
      └─ Used By
        └─ Equipment.EnchantWith(EnchantmentRecipe) : bool @06000F53
          └─ Equipment.Init(EquipmentItemMaster) : void @06000F46
            └─ Used By
              └─ ItemStack.Init(ItemMaster, int) : void @06000FC5
                └─ Used By
                  └─ ItemStack.Create(ItemMaster, int) : ItemStack @06000FC2
                    └─ Used By
                      └─ AbandonedItemsGenerator.Start() : void @0600135B
                        └─ BlacksmithPanel.CraftItem(Recipe) : void @06002649
                          └─ BreakableObjectBehaviour.DropObjects() : void @06001373
                            └─ Chest.Fill() : void @060013C8
                              └─ Chest.GenerateChestContent(ItemMaster.Culture) : void @060013CA
                                └─ Chest.GenerateFixedDrop() : void @060013C9
                                  └─ Chest.LoadFromSave(ChestSaveInfo) : void @060013E4
                                    └─ ChristmasTreeBehaviour.OnTriggerEnter2D(Collider2D) : void @06002E59
                                      └─ DesertBossBehaviour.DropDesertBossItemsOnDeath() : void @060015C1
                                        └─ Enemy.DropItemsOnDeath() : void @060016B1
                                          └─ HawkerPanel.OnAcceptCraft() : void @0600230E
                                            └─ HeroMerchant.DropRandomItem() : void @06001BB0
                                              └─ HeroMerchant.Update() : void @06001BB5
                                                └─ HeroMerchantController.Init() : void @06001C32
                                                  └─ HeroMerchantInventory.SetInitialEquippedItems() : void @0600276E
                                                    └─ HeroMerchantInventory.SetInitialItems() : void @0600276D
                                                      └─ HUDDebug.GiveItemToWill(ItemMaster, bool) : void @06003283
                                                        └─ HUDDebug.GiveWeaponToWill(string, bool, int) : void @06003284
                                                          └─ ItemDrop.SpawnItem(string, Vector3, float, Transform) : ItemStack @06001CD5
                                                            └─ ItemStack.Clone(Transform, int, bool) : ItemStack @06000FD5
                                                              └─ MoonlighterConsoleDebugPanel.OnAdditionalPotion() : void @060021A4
                                                                └─ PostmanInteractable.GiveWillBackerWeapon() : void @06002EFE

```

Item creation call analysis

And I saw these two methods in a class named `HUDDebug`. They have curious names `GiveItemToWill` and `GiveWeaponToWill`. If my guess is correct, there's a debug UI somewhere in the game that allows spawning items.



```

ShowEnemiesInfo() : void @06003292
SpawnEnemy(string) : void @06003290
SpawnEnemy(string, GameObject) : void @06003291
Start() : void @06003273
Update() : void @06003275
UpgradeShopTo(int) : void @0600327F
Instance : HUDDebug @170003BE
  accum : float @040028B7
  allowDrag : bool @040028B4
  areEnemiesLoaded : bool @040028D4
  color : Color @040028B9
  consoleDebugPanel : MoonlighterConsoleDebugPanel @040028F

```

HUDDebug Class

It has a `Start()` method, we can analyze it to see what calls it.

Seems like nothing calls it, same with `Awake()`.

```

Analyzer
  ▲ HUDDebug.Start() : void @06003273
    🔍 Used By
    ▶ 🔍 Uses
  ▲ HUDDebug.Awake() : void @06003272
    🔍 Used By
    ▶ 🔍 Uses

```

HUDDebug analysis

These are unity methods. According to this video

[https://unity3d.com/learn/tutorials/topics/scripting/awake-and-start:](https://unity3d.com/learn/tutorials/topics/scripting/awake-and-start)

1. `Awake()` : Called first even if the script is not enabled. Used for initialization.
2. `Start()` : Called only once after awake and before update if the script is enabled.
3. `Update()` : Called after the script is enabled and can be called multiple times.

These are called by the engine, so the `Analyze` tab will not have the chain.

Enabling Debug Mode

Inside `Update()` we can see:

```
73 // Token: 0x06003275 RID: 12917 RVA: 0x00145E94 File Offset: 0x00144094
74 private void Update()
75 {
76     if (GameManager.IsDebugEnabled)
77     {
78         if (ShopManager.Instance && ShopManager.Instance.isWillInShop && !ShopManager.Instance.IsShopClosed)
79         {
80             this.rectShopVisitors = new Rect((float)(Screen.width - 200), 0f, 200f, 600f);
81         }
82         this.accum += Time.timeScale / Time.deltaTime;
83         this.frames++;
84         this.SetScaleFactor();
85         if (this._showDebugInfo != GameManager.Instance.debug)
86         {
87             this._showDebugInfo = GameManager.Instance.debug;
88             if (this._showDebugInfo)
89             {
90                 base.StartCoroutine("FPS");
91             }
92             else
93             {
94                 base.StopCoroutine("FPS");
95             }
96         }
97     }
98 }
```

HUDDebug.Update()

Pay attention to line 76. There's an `if` condition that enables everything. We can analyze

`IsDebugEnabled`:

```
47
48 // Token: 0x170002DC RID: 732
49 // (get) Token: 0x06001F24 RID: 7972 RVA: 0x0001499B File Offset: 0x00012B9B
50 public static bool IsDebugEnabled
51 {
52     get
53     {
54         return Application.isEditor || Debug.isDebugBuild || Constants.GetBool("developerEnabled");
55     }
56 }
57
58 // Token: 0x06001F25 RID: 7973 RVA: 0x000DF47C File Offset: 0x000DD67C
59 public void UnloadMainMenuScene()
```

110 %

Analyzer

- GameManager.IsDebugEnabled : bool @170002DC
 - get
 - Used By
 - Uses

IsDebugEnabled analysis

Woot. It seems like we found it. Now we need to modify this to only return `true`.

Edit Method Body - get_IsDebugEnabled() : bool @06001F24

Instructions Locals Exception Handlers

Body Type IL Code

☐ Keep Old MaxStack ☒ Init Locals Header RVA 0x1499B Header Offset 0x12B9B

Index	Offset	OpCode	Operand
0	0000	call	bool [UnityEngine.CoreModule]UnityEngine.Application::get_isEditor()
1	0005	brtrue	7 (0020) ldc.i4.1
2	000A	call	bool [UnityEngine.CoreModule]UnityEngine.Debug::get_isDebugBuild()
3	000F	brtrue	7 (0020) ldc.i4.1
4	0014	ldstr	"developerEnabled"
5	0019	call	bool Constants::GetBool(string)
6	001E	br.s	8 (0021) ret
7	0020	ldc.i4.1	
8	0021	ret	

IL instructions for get_IsDebugEnabled

To return true, we need to return 1. We can delete lines 0 to 6 and only keep lines 7 and 8:

```
7: ldc.i4.1    // push 1 to the stack
8: ret        // return
```

Highlight lines 0 to 6 and press delete (or use the context menu) to remove them. Press to get the modified C# code.

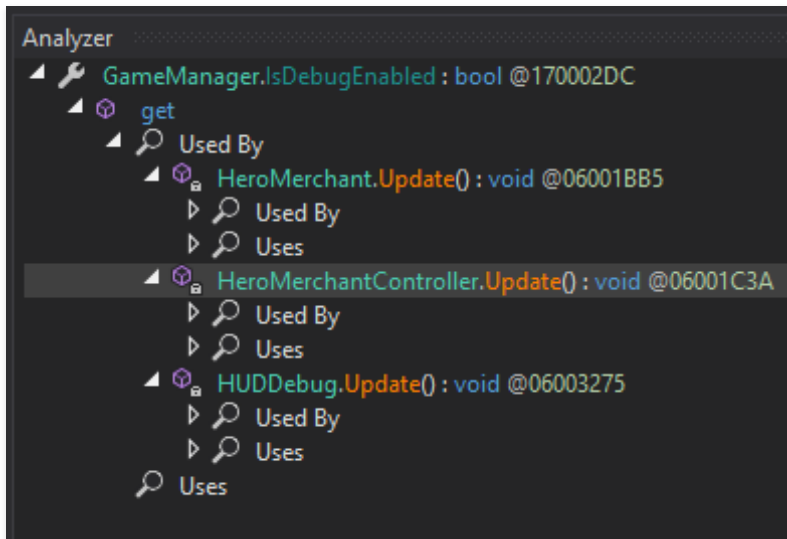

```
48 // Token: 0x170002DC RID: 732
49 // (get) Token: 0x06001F24 RID: 7972
50 public static bool IsDebugEnabled
51 {
52     get
53     {
54         return true;
55     }
56 }
```

Modified get_IsDebugEnabled

This enables debug mode for every run. But what else is out there?

Debug Shortcuts

Let's go back to the analysis results for `GameManager.IsDebugEnabled`



IsDebugEnabled analysis

We have already looked at `HUDDDebug.Update`, now we look at the other two.

`HeroMerchant.Update()` has some shortcuts.

```
// Token: 0x06001BB5 RID: 7093 RVA: 0x000C42D8 File Offset: 0x000C24D8
private void Update()
{
    if (GameManager.IsDebugEnabled)
    {
        if (Input.GetKeyDown(99) && Input.GetKey(304))
        {
            this.RepositionInShopOrDungeon();
        }
        if (Input.GetKeyDown(104) && Input.GetKey(304))
        {
            ItemStack itemStack = this.heroMerchantInventory.GetEquippedItemByType(HeroMerchantInventory.EquipmentSlot.Potion);
            if (itemStack && itemStack.FreeStack > 0)
            {
                itemStack.Quantity++;
                this.heroMerchantInventory.SetEquippedItemByType(itemStack, HeroMerchantInventory.EquipmentSlot.Potion);
            }
            else if (!itemStack)
            {
                ItemMaster itemByName = ItemDatabase.GetItemByName("HP Potion IV");
                itemStack = ItemStack.Create(itemByName, 1);
                this.heroMerchantInventory.SetEquippedItemByType(itemStack, HeroMerchantInventory.EquipmentSlot.Potion);
            }
            else
            {
                ItemMaster itemByName2 = ItemDatabase.GetItemByName("HP Potion IV");
                itemStack = ItemStack.Create(itemByName2, 1);
                if (this.heroMerchantInventory.TryAddItem(itemStack, 0) && itemStack.Quantity == 0)
                {
                    Object.Destroy(itemStack.gameObject);
                }
            }
        }
        if (Input.GetKeyDown(113) && Input.GetKey(304))
        {
            DungeonManager instance = DungeonManager.Instance;
            if (instance != null)
            {

```

HeroMerchant.Update()

KeyCodes

[Input.GetKeyDown](#) detects when a key is pressed and released. The parameter to the method is an enum of type [KeyCode](#) or a string. It took me a while to find out the associated numbers.

I found it in the decompiled code at:

- <https://github.com/jamesjlinden/unity-decompiled/blob/master/UnityEngine/UnityEngine/KeyCode.cs>

[Here's a local copy](#), in case the repository is taken down.

It's based on ASCII-Hex decimal values with extra keys (e.g. gamepad) in the end.

Now we can decipher some debug shortcuts:

```
if (Input.GetKeyDown(104) && Input.GetKey(304))
{
    ItemStack itemStack = this.heroMerchantInventory.GetEquippedItemByType(HeroMerchantInventory.EquipmentSlot.Potion);
    if (itemStack && itemStack.FreeStack > 0)
    {
        itemStack.Quantity++;
        this.heroMerchantInventory.SetEquippedItemByType(itemStack, HeroMerchantInventory.EquipmentSlot.Potion);
    }
    else if (!itemStack)
    {
        ItemMaster itemByName = ItemDatabase.GetItemByName("HP Potion IV");
        itemStack = ItemStack.Create(itemByName, 1);
        this.heroMerchantInventory.SetEquippedItemByType(itemStack, HeroMerchantInventory.EquipmentSlot.Potion);
    }
    else
    {
        ItemMaster itemByName2 = ItemDatabase.GetItemByName("HP Potion IV");
        itemStack = ItemStack.Create(itemByName2, 1);
        if (this.heroMerchantInventory.TryAddItem(itemStack, 0) && itemStack.Quantity == 0)
        {
            Object.Destroy(itemStack.gameObject);
        }
    }
}
```

Potion Shortcut

- 104 : H
- 304 : LeftShift

Note we can also change the item granted with anything we want.

```
if (Input.GetKeyDown(98) && Input.GetKey(304))  
{  
    this.TeleportToFloorEnd();  
}
```

We can teleport to the last room of any dungeon floor:

- 98 : B
- 304 : LeftShift

And so on.

Controller Shortcuts

`HeroMerchantController.Update()` defines controller shortcuts.

```
if (GameManager.IsDebugEnabled)  
{  
    if (Input.GetKeyDown(121))  
    {  
        this.MoveAutomaticallyToPoint(base.transform.position + Vector3.right * 100f, 2f, true, false);  
    }  
    if (Input.GetKey(306) && (Input.GetKeyDown(273) || Input.GetKeyDown(274)))  
    {  
        this.DisableVerticalMoveInput();  
    }  
    if (Input.GetKey(306) && (Input.GetKeyDown(276) || Input.GetKeyDown(275)))  
    {  
        this.DisableHorizontalMoveInput();  
    }  
    if (Input.GetKeyDown(48))  
    {  
        this.EnableMoveInput();  
    }  
}
```

Controller shortcuts

Save/Reset Shortcuts

There are a couple of more shortcuts inside `HUDDebug.Update()`:

```
if (Input.GetKeyDown(115) && Input.GetKeyDown(304))
{
    Debug.Log("Saving game with shrotcut");
    GameManager.Instance.SaveAll();
}
if (Input.GetKeyDown(114) && Input.GetKeyDown(304))
{
    Debug.Log("Resetting save game with shrotcut");
    GameManager.Instance.HardResetSave();
    GameManager.Instance.SerializeGameSlot();
}
```

Save/reset shortcuts

- Save: `LeftShift` + `S`
- Reset: `LeftShift` + `R`

Enabling Debug HUD

We still need to enable the HUD. It must have a shortcut key. Searching the internet tells us it's `Tab`.

See this page about a CheatEngine trainer:

- <http://fearlessrevolution.com/viewtopic.php?p=47682#p47682>

How can we find it ourselves? Back inside `HUDDebug.Update()` we can see a block that enables and disables `consoleDebugPanel`:

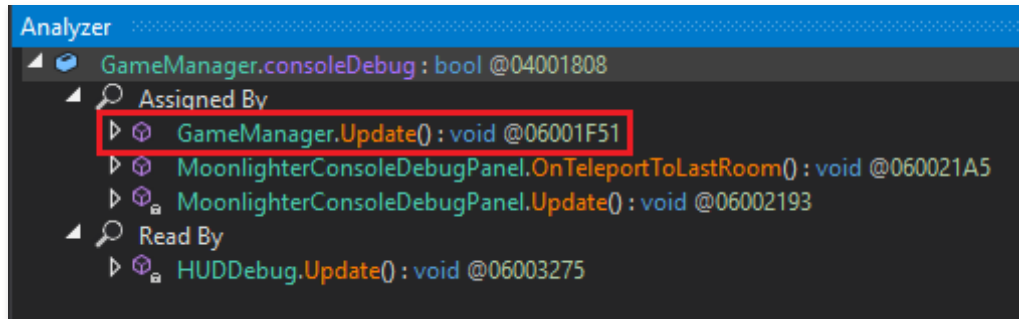
```

if (this._showConsoleDebugInfo != GameManager.Instance.consoleDebug && this.consoleDebugPanel)
{
    this._showConsoleDebugInfo = GameManager.Instance.consoleDebug;
    if (this._showConsoleDebugInfo)
    {
        this.consoleDebugPanel.Enable();
    }
    else
    {
        this.consoleDebugPanel.Disable();
    }
}

```

Enabling and disable consoleDebugPanel

Inside the `if` we can see that `GameManager.Instance.consoleDebug` is referenced. Let's analyze that.



consoleDebug analysis

Double-click on `GameManager.Update()`:

```

710     public void Update()
711     {
712         Shader.SetGlobalFloat("_UnscaledTime", Time.unscaledTime);
713         if (this._startSequenceMovie)

```

```

714     {
715         this.CheckVideoSequenceSkip();
716     }
717     if (Input.GetKeyDown(9))
718     {
719         this.debug = !this.debug;
720         if (this.debug || this.isTest)
721         {
722             Cursor.visible = true;
723         }
724         else
725         {
726             Cursor.visible = false;
727         }
728     }
729     if (GUIManager.Instance.input.ButtonRightStick.IsPressed)
730     {
731         this._lapse += Time.deltaTime;
732         if (this._lapse > 1f)
733         {
734             this._lapse = 0f;
735             this.consoleDebug = true;
736         }
737     }
738     if (GUIManager.Instance.input.ButtonRightStick.WasReleased)
739     {
740         this._lapse = 0f;
741     }
742     if (Input.GetKeyDown(48))
743     {
744         this.SetGameSpeed(Constants.GetFloat("defaultGameTimeScale"));
745     }
746 }

```

GameManager.Update()

We can see it's triggered by holding the `ButtonRightStick` for a second (I think). A bit further up we can see the equivalent keyboard shortcut.


```
if (Input.GetKeyDown(9))
```

It's the `Tab` key which confirms what we found online. We can press `Tab` in the game to enable debug HUD.



Debug HUD

And it's quite extensive.

Lessons Learned

We learned:

- How to use dnSpy's Analyze feature to track variables/methods/etc.
- How to enable debug mode.
- Shortcut keys for various things such as potions.
- Shortcut key to enable the debug HUD.

My next plan was to use Cheat Engine to make cheats and enable debug HUD. But it's already done. In the next part, I will write my thoughts about some questions that I was asked about such cheating. Maybe after that, I will return and do the Cheat Engine part but I will need to re-learn the tool again.

Posted by Parsia • Jan 29, 2019 • Tags: [Moonlighter](#) [dnSpy](#)

[Cheating at Moonlighter - Part 2 - Changing Game Logic with dnSpy](#)

[Cheating at Moonlighter - Part 4 - Defense](#)

0 Comments

Parsiya

1 Login ▾

♥ Recommend

🐦 Tweet

📌 Share

Sort by Best ▾



Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS ?



Name

Be the first to comment.

✉ Subscribe

🗣 Add Disqus to your site

🔒 Disqus' Privacy Policy

DISQUS

Copyright © 2019 Parsia - [License](#) - Powered by [Hugo](#) and [Hugo-Octopress](#) theme.