Team: WhatInTheHell

Members: Michael Chacko
Kevin Gamage
Kevin Kuriachan
Nabeel Sheikh

https://github.tamu.edu/kevinkuriachan/DLL_Injection_Hooking_Pwn_Island

Online / on the server:

Until the Cows Come Home

The main exploit our team developed to obtain this flag was a fly hack. When we first began trying to manipulate game data using Cheat Engine, we found that the mana and health values were stored server-side and we could not easily manipulate their values. However, we were able to learn that our player's coordinates could be easily changed without verification from the server.

Developing a fly hack with the aid of this online tutorial and commenter drew6017, it proved to be a fun and challenging experience.

Using Visual Studio, we developed a dll file with the code necessary to manipulate the player coordinates with the WASD and CTRL & SHIFT keys. The dll file creates a new thread which gets the handle address of PwnAdventure's GameLogic.dll file.

```
uintptr_t gamelogicbase = (uintptr_t)GetModuleHandleW(L"GameLogic.dll");
```

Using Cheat Engine, we inject the DLL into the game. The F key toggles the hack, while SHIFT and CONTROL modify the desired Z value of the player's position.

```
for (;;) {

    if (GetAsyncKeyState('F') & 1) {
        flying = !flying;
        std::cout << "Flying toggled: " << (flying ? "ON" : "OFF") << std::endl;
        prevz = playerLoc->z;
    }

    if (flying) {
        if (GetAsyncKeyState(VK_SHIFT) & 0x8000) {
            prevz += INC_AMOUNT;
        }
        else
        {
            if (GetAsyncKeyState(VK_CONTROL) & 0x8000) {
                prevz -= INC_AMOUNT;
            }
        }
        playerLoc->z = prevz;
```

The playerLoc variable, a 3d vector, was found at the address of the GameLogic.dll file offset by 0x00097D7C. By modifying this variable, we are able to easily modify the player's location. Once the hack was perfected, we flew to the island in the corner of the online game server and killed the King Cow!

Offline:

The approach:

Since there is no server to perform checks and keep track of true values, any modifications to game instructions on the server side are valid. The approach to solving these problems offline was to do a DLL injection.

The DLL to be injected can be found at this repository:
https://github.tamu.edu/kevinkuriachan/DLL_Injection_Hooking_Pwn_Island

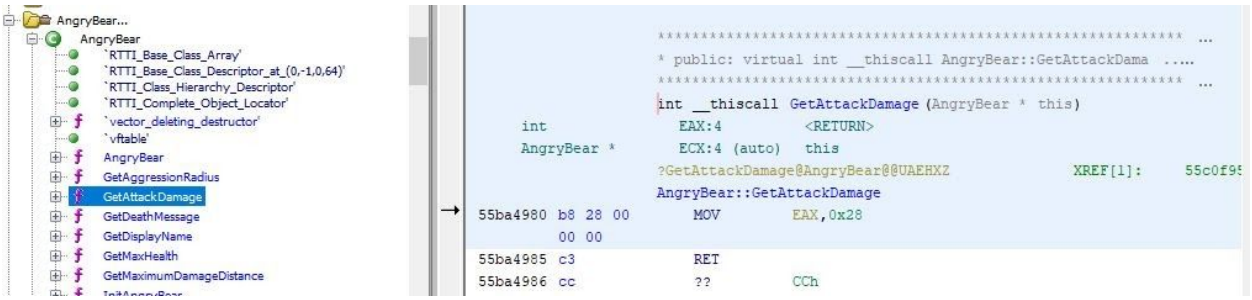
Cheat Engine 7.0 allows us to attach to a process and inject a DLL file into it. The DLL will start a thread that is a child of the game process and therefore has access to all the memory that the game has.

Line 249 of dllmain.cpp is the thread started by the DLL injection. The win32 GetModuleHandle function provides the base address of a DLL file. In our case, the GameLogic.dll file:

```
HMODULE moduleHandle = GetModuleHandle(L"GameLogic.dll");
LPUINT moduleBase = (LPUINT) moduleHandle;
```

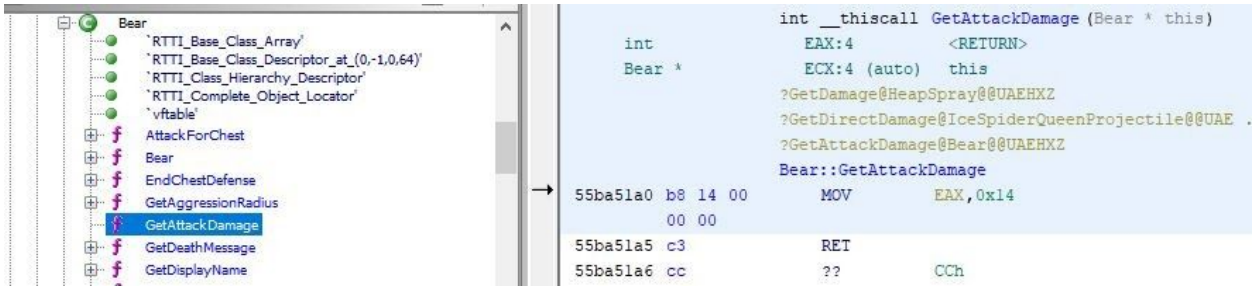
Ghidra reveals many classes in GameLogic.dll and each function can be located in memory by the offset from the base address. These offsets are also shown by Ghidra. The reason addresses have to be calculated as an offset from the base address each time is because Windows uses Address Space Layout Randomization (ASLR) to randomize the virtual address space of a process each time it is started.

Inspection in Ghidra showed enemy classes such as the Bear, AngryBear, GiantRat, and MadCows for example. Each of these classes has a GetAttackDamage() method which returns some number. If these are overridden to return 0, then the enemy does no damage.



Unbearable Revenge

The goal in Unbearable Revenge is to stay within a radius of the chest for five minutes and defend oneself while bears and angry bears attack.

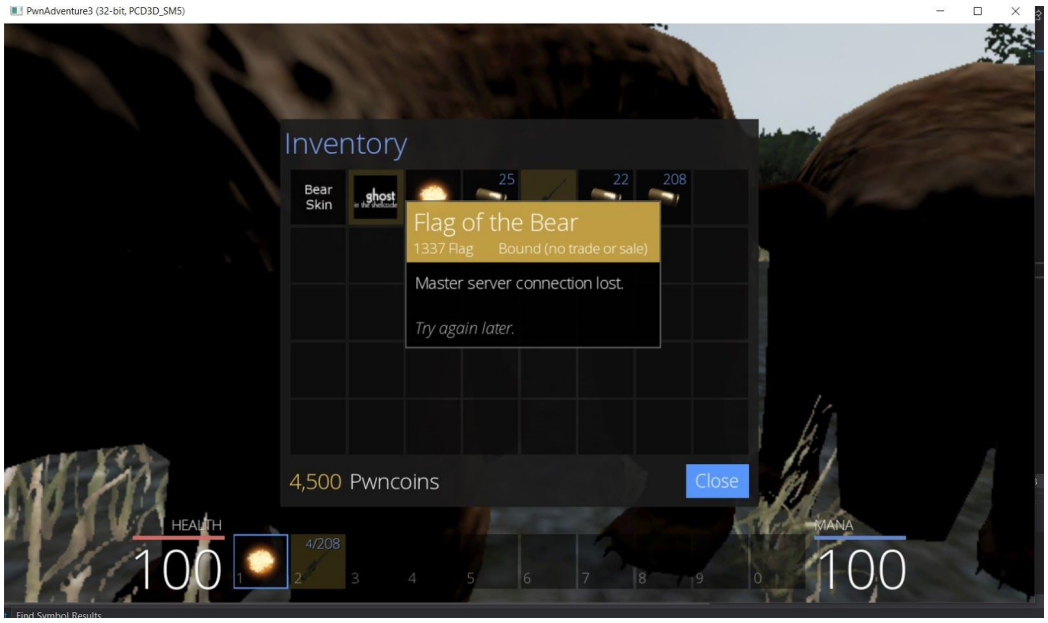


The offsets of each enemy's attack damage is shown by Ghidra. For example, the return value of Bear::GetAttackDamage() is at offset 0x51a1 and AngryBear::GetAttackDamage() is at 0x4981. These values can be replaced with a zero as follows:

```
*(BYTE*)addr = 0x00;
```

where addr is baseAddress + offset.

This is performed for both the bear and angry bear in order to take no damage from then, thus solving Unbearable Revenge:



Pirate's Treasure

In Pirate's Treasure, the goal is to guess the key that solves the problem. Inspection of game logic shows that there is a GameAPI.KeyVerifier::VerifyKey() function.

The exploit in this case, was to "hook" the function and redirect it's control flow. This is done by replacing some instruction at the start of the function call with a jump into custom code. The DLL injection created by our team does this with

BOOL HookFunction(void* addrToHook, void* myFunction, int len)

Starting with addrToHook, a len number of NOP instructions are written. Then, addrToHook gets a jump written to it that redirects control flow to myFunction.

The function is called with arguments to the address directly after EBP is pushed by the verification function.

```
DWORD verifyJumpBackAddr;
void __declspec(naked) verfiyKeyDetour()
{
	__asm {
		mov al, 0x1;
		jmp[verifyJumpBackAddr]
	}
}
```

The detour will then transfer control back to an address of our choosing. In this case, it is transferred back to the epilogue of the function we hooked. Effectively, the VerifyKey function is redirected to our function which places a 1 into the return value and then goes back to the function return.

GameLogic.KeyVerifier::VerifyKey	55	push	ebp
GameLogic.KeyVerifier::VerifyKey+1	8B EC	mov	ebp,esp
GameLogic.KeyVerifier::VerifyKey+3	E9 E865D48B	jmp	blockyOpen.verfiyKeyDetour
GameLogic.KeyVerifier::VerifyKey+8	90	nop	
GameLogic.KeyVerifier::VerifyKey+9	90	nop	
GameLogic.KeyVerifier::VerifyKey+A	90	nop	

GameLogic.KeyVerifier::VerifyKey+2AE	90	nop	
GameLogic.KeyVerifier::VerifyKey+2AF	90	nop	
GameLogic.KeyVerifier::VerifyKey+2B0	90	nop	
GameLogic.KeyVerifier::VerifyKey+2B1	90	nop	
GameLogic.KeyVerifier::VerifyKey+2B2	8B E5	mov	esp,ebp
GameLogic.KeyVerifier::VerifyKey+2B4	5D	pop	ebp
GameLogic.KeyVerifier::VerifyKey+2B5	C3	ret	

The function after exploit

blockyOpen.verfiyKeyDetour()				
blockyOpen.verfiyKeyDetour	B0 01	mov	al,01	1
blockyOpen.verfiyKeyDetour+2	FF 25 8C432C07	jmp	dword ptr [blockyOpen.verifyJumpBackAddr]	->GameLogic.!

Detour function

Now, any input will solve the problem.