Detour Hack Project

Shawn Hagler
EEL-4347
Florida State University
Panama City, United States
sth20u@fsu.edu

Jaehyun Lee
EEL-4347
Florida State University
Panama City, United States
jl21bd@fsu.edu

Noah Fielder
EEL-5348
Florida State University
Panama City, United States
nlf21@fsu.edu

ABSRACT

This paper presents a project that uses Microsoft's Detour library and virtual metthod table (VMT) hooking to create a hack for the classic video game Battlefield 1942. The Detour library is used to intercept calls to the game's functions, allowing us to modify the game's behavior. VMT hooking is used to modify the game's classes and functions, allowing us to add custom behavior to the game. By combining these two methods, we are able to implement a hack that allows players to gain an advantage in the game by manipulating the game state. The hack is tested on a Windows 10 system and the results are discussed. The paper ends by outlining future work and potential applications of the hack.

I. INTRODUCTION

In the past few decades, video games have become an increasingly popular form of entertainment around the world. As the technology used to create video games has become more sophisticated, the difficulty of cheating in them has grown as well. The goal of the project presented in this paper is to showcase basic examples of video game cheating in the classic video game Battlefield 1942. The choice of this video game was determined as the game used to be very popular but is not played that often and online multiplayer servers have been disbanded. This makes the game an ideal choice for creating and publishing a video game cheat as it is not very likely to have an adverse affect on other players.

II. EXISTING WORKS

A. Microsoft Detour Library

Microsoft's Detour library is a software development tool released by Microsoft Research in 2002; initial publication of their paper was in 1999. The library is used to intercept calls to an application's API, allowing developers to modify the behavior of the application. The Detour library provides an easy-to-use interface for developers to create and manage application hooks. It is composed of two parts: the Detour library, which provides the interface for creating and managing hooks, and the Detour library extension (Detex), which proides additional features such as logging and debugging. The Detour library is widely used in software development to analyze and modify the behavior of applications. It has been used to create game hacks, security tools, automated testing tools, and more.

III. IMPLEMENTATION

A. Dynamic Link Library Injection

Dynamic Link Library (DLL) injection is one of the techniques used in our project to create the hack for Battlefield 1942. DLL injection is a process in which a DLL is loaded into the address space of a process. This allows the DLL to access the process's memory and resources and execute arbitrary code. DLL injection can be used to modify the behavior of the process by loading custom code into the process's address space. In our project, we use DLL injection to load a DLL containing our custom code into the game process. This allows us to intercept calls to the game's API and modify the game's behavior.

B. Function Detouring

Function detouring is another technique used in our project to create the hack for Battlefield 1942. Function detouring is the process of intercepting the execution of a function call to modify its behavior. This is done by creating a new function that is called instead of the original one. This new function can then be used to modify the behavior of the original function. This process "detours" or "reroutes" the execution of code in the target process.

To create a function detour using Microsoft's Detour library, the following steps must be taken:

1) Create a new function that will be used as the detour. This new function must have the same signature as the original function it is replacing.
2) Call the **DetourFunction** function from the Detour library to create a detour from the original function to the new function.
3) Use the **DetourRemove** function from the Detour library to remove the detour when it is no longer needed.

By using the Detour library, it is possible to intercept and modify the behavior of any function in a program. This can be used to create video game hacks that give the user an advantage in the game.

C. Virtual Method Table Hooking

Virtual method table (VMT) hooking is another technique used in our project to create the hack for Battlefield 1942. VMT hooking is used to modify the behavior of a data structure within a process by intercepting class to functions in the data structure. It is used to modify the behavior of a class without modifying the source code of the class. The process of VMT hooking involves replacing the pointer to a particular function in the VMT with a pointer to a new function that will be used instead of the original one. This allows the new function to be called instead of the original one. The usefulness of this function over using Microsoft's Detour library is that it is easily used on functions within C++ object-oriented data structures that output a virtual method.

To create a VMT hook, the following steps must be taken:

1) Find the VMT of a particular data structure. This can be done by searching for the start of the VMT in memory.
2) Find the function in the VMT that needs to be replaced.
3) Create a new function that will be used as the replacement. This new function must have the same signature as the original one.
4) Replace the pointer to the original function in the VMT with a pointer to the new function.

By using VMT hooking, it is possible to modify the behavior of a data structure without modifying the source code. This can be used to create video game hacks that give the user an advantage in the game.

D. Reverse Engineering

Reverse engineering a video game typically involves disassembling the game's executable file and examining the machine code to understand how the game works. This can be done using a tool called a disassembler, which converts the machine code into human-readable assembly language. This process is crucial in creating the video game hack for Battlefield 1942 as the contents of important data structures and their offsets must be known. This process is also used to identify the game's virtual method tables.

As described in the above section, virtual tables are arrays of function pointers that the game uses to call methods in an object-oriented program. By examining virtual tables, it is possible to determine how the game's objects are organized and how they interact with each other. Once the virtual tables have been identified, the next step is to locate the addresses where the game's functions and data structures are stored in memory. This can be done by searching the disassembled code for references to these addresses, or by using a tool called a debugger to monitor the game as it is running and watch how it accesses these addresses.

Overall, the process of reverse engineering a video game involves disassembling the game's code, identifying its virtual tables and data structures, and locating the addresses of its functions and data structures in memory. This can provide insight into how the game works and allow the user to modify its behavior.

E. Code Example

The following is a simple code example for function detouring:

```
// `source_address`
//   - the memory address where we want
//     to place our `jmp` instruction
// `dest_address`
//   - the memory address of the attacker's code
// `length`
//   - the number of bytes used by the instruction
//     we are overwriting
void
Detour(void* src_addr, void* dest_addr, int length) {
    // The smallest relative `jmp` in x86 is 5 bytes
    if (length < 5) return;

    // Take control of the memory region
    // we're attacking.
    DWORD old_prot = {};
    VirtualProtect(src_addr, length,
                   PAGE_EXECUTE_READWRITE,
                   &old_prot);

    // Overwrite the memory address with the `nop`
    // (0x90) instruction. This is not necessary
    // but is a nice failsafe measure and provides
    // a nice point of reference when debugging.
    memset(src_addr, 0x90, length);

    // Calculate the relative address between the
    // destination address and the source address
    // by taking the difference minus the length.
    // This gives us the relative offset from the
    // last byte that we have overwrote to the
    // address of the attack region. This is
    // neccessary as we are performing a relative
    // `jmp` not an absolute `jmp` so this must
    // be calculated at runtime.
    DWORD rel_addr = ((DWORD)dest_addr -
                      (DWORD)src_addr) - length;

    // Overwrite the source address with the relative
    // `jmp` instruction (0xE9).
    *(BYTE*)src_addr = 0xE9;

    // Add one byte to the source address so that
    // we can write the relative offset.
    *(DWORD*)((DWORD)src_addr + 1) = rel_addr;

    // Restore the memory region protections.
    DWORD new_prot = {};
    VirtualProtect(src_addr, length, old_protection,
                   &new_prot);
}
```

The following is a simple code example for VMT hooking:

```
    // Define the original virtual method
    class OriginalVTable {
        public:
            virtual void do_something();
    };

    // Define a hook class that will replace
    // the original method with a new implementation
    class Hook : OriginalVTable {
        public:
            virtual void do_something() {
                /* new implementation */
            }
    };

    int main() {
        // Create an instance of the original class
        OriginalVTable* orig = new OriginalVtable();

        // Create a pointer to the original VMT
        void** vmt = *reinterpret_cast<void***>(orig);

        // Replace the original VMT with the
```

```
        // hooked VMT
        *vmt = *reinterpret_cast<void**>(new Hook());

        // Call the original method, which will
        // now be redirected to the hook method
        orig->do_something();

        return 0;
    }
```

IV. Results

After the hack DLL is injected, we spawn and attach a debug console for printing out values and logging for ease of debugging. Figure 1 below showcases this in action.

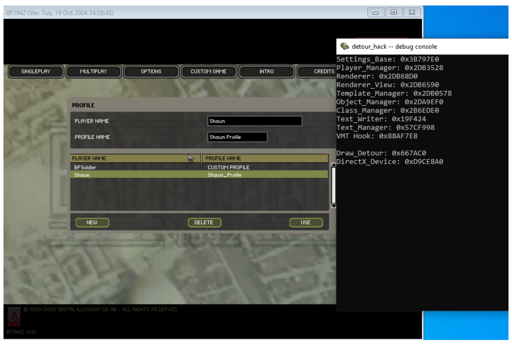

Fig. 1. Cheat initialization.

Fig. 2. Cheat radar.

Figure 2 showcases the main goal of this video game hack where we display enemies location on the game's mini-map that would normally only show teammate's locations. This would potentially give an advantage to the cheater within the game. It is not uncommon for players to use hacks or other unauthorized methods to gain an advantage in online multiplayer games, although this is typically against the rules and can result in penalties or bans. Figure 3 just simply provides a zoomed image of the feature for better view.

V. Conclusion

This paper presented a project that combined Microsoft's Detour library and virtual method table (VMT) hooking to create a hack for the classic video game Battlefield 1942. By intercepting calls to the game's functions and modifying the game's classes and functions, we were able to create a hack that allowed players to gain an advantage in the game by manipulating the game state.

Fig. 3. Cheat radar zoomed in.

Overall, this project demonstrates the power of combining Microsoft's Detour library and VMT hooking to create powerful hacks that can give players an advantage in a game. This project can be used as a starting point for creating similar hacks for other games, or for other applications that require function interception and modification.

A. Future Work

Future work on this project should focus on completing the implementation for aimbot and visual ESP features. Aimbot is a feature that allows players to automatically aim and shoot at enemies, while visual ESP features provide players with a full view of the game world, allowing them to see enemy positions and other information.

To implement these features, Microsoft's Detour library and VMT hooking will need to be used to intercept calls to the game's functions and modify the game's data structures and functions. The aimbot feature will require additional code to track and aim at enemy players, while the visual ESP feature will require code to render a 3D view of the game world.

Once these features have been implemented, they can be tested in-game to ensure that they are working correctly. This will provide a valuable insight into the effectiveness of the hack and can help to identify potential areas for improvement.

References

[1] Hunt, "Detours: Binary Interception of Win32 Functions", *Third USENIX Windows NT Symposium*, https://www.microsoft.com/en-us/research/publication/detours-binary-interception-of-win32-functions/, July 1999

[2] Niemand, "Virtual Method Table Hooking Explained," niemand.com.ar, 30 Jan. 2019. [Online]. Available: https://niemand.com.ar/2019/01/30/virtual-method-table-hooking-explained/. [Accessed: 01 Dec. 2022].