```
add     rcx, 60h ;
mov     rdx, rax
call    cs:sf::RenderTarget::clear(sf::Color
mov     eax, [rbp+3A0h+counter_1]
inc     eax
mov     eax, eax
mov     edx, eax        ; unsigned __int64
mov     rcx, [rbp+3A0h+p_vector] ; this
call    Controller::moveBetweenWaves(unsigned
mov     eax, [rbp+3A0h+counter_1]
mov     rcx, [rbp+3A0h+p_vector]
add     rcx, 680h
mov     edx, eax
call    std::vector<std::unique_ptr<WaveInter
mov     rcx, rax
call    std::unique_ptr<WaveInterface,std::de
mov     [rbp+3A0h+p_wave], rax
mov     rax, [rbp+3A0h+p_wave]
mov     rax, [rax]
mov     rcx, [rbp+3A0h+p_wave]
call    qword ptr [rax]
mov     rax, [rbp+3A0h+p_vector]
add     rax, 8
mov     [rbp+3A0h+p_wave], rax
mov     ecx, [rbp+3A0h+counter_1]
mov     rdx, [rbp+3A0h+p_vector]
add     rdx, 680h
mov     [rbp+3A0h+counter], rdx
mov     edx, ecx
mov     rcx, [rbp+3A0h+counter]
call    std::vector<std::unique_ptr<WaveInter
mov     rcx, rax
call    std::unique_ptr<WaveInterface,std::de
mov     rcx, [rbp+3A0h+p_wave]
mov     rdx, rcx
mov     rcx, rax
call    WaveInterface::getWaveStage(std::vect
mov     eax, [rbp+3A0h+counter_1]
mov     rcx, [rbp+3A0h+p_vector]
add     rcx, 248h
mov     edx, eax
call    Level::runNewLevel(unsigned __int64)
mov     rcx, [rbp+3A0h+p_vector]
mov     [rcx+6A8h], eax
mov     eax, [rbp+3A0h+counter_1]
inc     eax
mov     [rbp+3A0h+counter_1], eax
mov     rax, [rbp+3A0h+p_vector]
add     rax, 8
xor     edx, edx
mov     rcx, rax
call    std::vector<std::shared_ptr<GameObjec
```

Behind Enemy Lines Reverse Engineering C++ in Modern Ages

Gal Zaban
@0xgalz


```
    if (it == _sprites_map.end())
        return nullptr;
    return std::make_unique<sf::Sprite>
}
//-------------------------------------
// The follwing function returns the sc
//-------------------------------------
std::vector<std::unique_ptr<sf::Text>>
{
    return _scoreBoardTexts;
}
//-------------------------------------
// The function returns the font it upl
//-------------------------------------
sf::Font & ResourcesManager::getFont()
{
    return _m_font;
}
//-------------------------------------
// This function loads all of the sprit
// The needed sprites from a txt file
//-------------------------------------
void ResourcesManager::loadSprites()
{
    std::ifstream spritesLoader;
    spritesLoader.open("Sprites.txt");
    if (!spritesLoader.is_open())
        throw std::ios_base::failure("C
    std::string spriteName = "Resources
    int indx = 0;

    while (!spritesLoader.eof())
    {
        std::string theName;
        spritesLoader >> theName;
        spriteName += theName;
        std::unique_ptr<sf::Texture> te
        if (!text->loadFromFile(spriteN
            throw std::ios_base::failur
        _texture.emplace_back(std::move
        _sprites_map.emplace(theName, *
        spriteName = "Resources/images/
        ++indx;
    }

    spritesLoader.close();
```

id;

- Gal Zaban.
- Security Researcher.
- I break yo’ stuff and sew for fun and non profit.

Agenda

- C++ from a Reverse Engineer Perspective.
- Picking a Target.
- Winning.
- Putting it into Practice.
- Have fun! :)

Binary Creation

What if we would like to reverse the process?

?


Binary
Code

Reverse Engineering

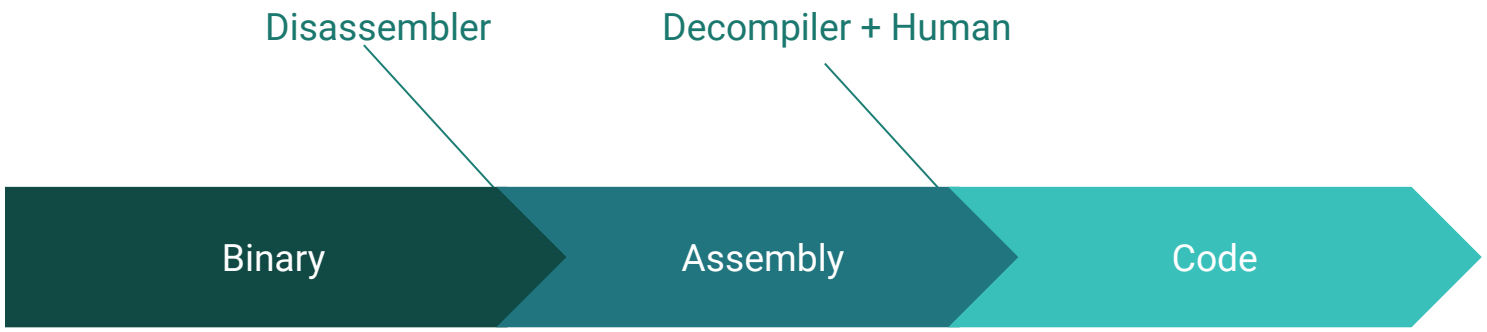
- Locals and Names don’t appear in the binary.
  - Unless the code was compiled in debug mode.
- Optimizations usually make my head hurt.
- Static vs. Dynamic reversing.
- Hardware vs. Software.
- And many more.

What is it good for?

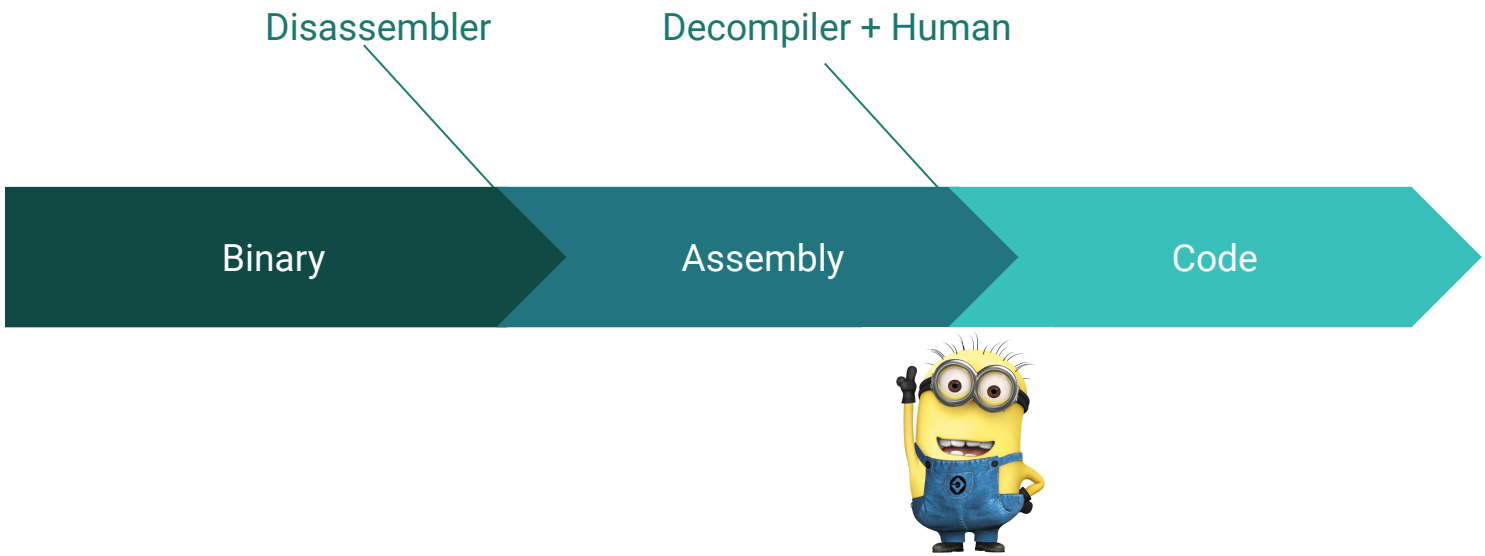
- Vulnerabilities
  - Look for bugs in a code and exploit them.

- Understand the logic of code/ algorithm.
  - Solve complex synchronization and bad optimizations.

The Reversed Steps



The Reversed Steps



Now, for a real example :)

What is the most effective way to learn Reversing of C++ code?


CHICKEN
INVADERS


209,605
11
0
8
15

Win Chicken Invader!

But there is a problem..

I am TERRIBLE at this game…
The only way I can win
is….**cheating**!!11

Objective:
Win Chicken Invaders
By Patching the Game!

Let’s Start!

What do we want?

- Make the game easier.
  - Make the Chicken Boss easier.
- Learn reverse engineering C++ on the way.

What do we have?

- The binary of the game.
- A tool for disassembling the code.
- Our knowledge in C++.
- Some knowledge in reverse engineering (?).

Our tools for Reverse Engineering

Tools we use- Disassemblers

Tools we use- Disassemblers

- IDA is our fav tool.

Assembly Basics (For MSVC Compiler)

- Important registers.
  - RAX/ EAX- stores the return value of functions.
  - RCX/ ECX- Stores pointers to objects.
- MSVC __fastcall calling convention uses Registers to pass Parameters
  - RCX, RDX, RDI, RSI, R8, R9
  - Anything else goes to the stack.

C++ Concepts in 60 seconds

Dynamic Object Creation

```
class Father1
{
public:
    Father1(int age) { _age = age; }
    virtual int GetAge() { return _age; }
    virtual void PrintAge() { cout << "Father1's age is: " << _age << endl; }
private:
    int _age;
};

int main()
{
    Father1* objA = new Father1(52);
```

Dynamic Object Creation

```
class Father1
{
public:
    Father1(int age) { _age = age; }
    virtual int GetAge() { return _age; }
    virtual void PrintAge() { cout << "Father1's age is: " << _age << endl; }
private:
    int _age;
};

int main()
{
    Father1* objA = new Father1(52);
```

```
push    8                   ; Size
call    operator new(uint)
```

Dynamic Object Creation

```
class Father1
{
public:
    Father1(int age) { _age = age; }
    virtual int GetAge() { return _age; }
    virtual void PrintAge() { cout << "Father1's age is: " << _age << endl; }
private:
    int _age;
};

int main()
{
    Father1* objA = new Father1(52);
```

```
push    8                   ; Size
call    operator new(uint)
```

```
call    j_gz_Father1_ctor
mov     [ebp+this_ptr_Father1], eax
```

Basic Constructors

- How can we recognize a constructor in Assembly?
  - Sets the vtable to the object first bytes (4/ 8 - x86/x64)
  - Sets the object members in the other offsets
  - In case of inheritance, also calling the father's constructor.
- As can be seen in the following example:

```
mov     eax, [ebp+p_this_object]
mov     dword ptr [eax], offset const Account::`vftable'
mov     eax, [ebp+p_this_object]
movsd   xmm0, [ebp+objects_member]
movsd   qword ptr [eax+8], xmm0
```

Basic Constructors

- How can we recognize a constructor in Assembly?
  - **Sets the vtable to the object first bytes (4/ 8 - x86/x64)**
  - Sets the object members in the other offsets
  - In case of inheritance, also calling the father's constructor.
- As can be seen in the following example:

```
mov     eax, [ebp+p_this_object]
mov     dword ptr [eax], offset const Account::`vftable'
mov     eax, [ebp+p_this_object]
movsd   xmm0, [ebp+objects_member]
movsd   qword ptr [eax+8], xmm0
```

Basic Constructors

- How can we recognize a constructor in Assembly?
  - Sets the vtable to the object first bytes (4/ 8 - x86/x64)
  - **Sets the object members in the other offsets**
  - In case of inheritance, also calling the father's constructor.
- As can be seen in the following example:

```
mov     eax, [ebp+p_this_object]
mov     dword ptr [eax], offset const Account::`vftable'
mov     eax, [ebp+p_this_object]
movsd   xmm0, [ebp+objects_member]
movsd   qword ptr [eax+8], xmm0
```

Virtual calls

```
mov     [ebp+this_object], eax
mov     eax, [ebp+this_object]
mov     edx, [eax]
mov     esi, esp
mov     ecx, [ebp+this_object]
mov     eax, [edx+4]
call    eax
```

Virtual calls

```
mov     [ebp+this_object], eax
mov     eax, [ebp+this_object]
mov     edx, [eax]
mov     esi, esp
mov     ecx, [ebp+this_object]
mov     eax, [edx+4]
call    eax
```

The virtual call

- The function that will be called will change on runtime

Virtual calls

```
mov     [ebp+this_object], eax
mov     eax, [ebp+this_object]
mov     edx, [eax]
mov     esi, esp
mov     ecx, [ebp+this_object]
mov     eax, [edx+4]
call    eax
```

Move the virtual function to EAX

- The function that will be called will change on runtime

Virtual calls

```
mov     [ebp+this_object], eax
mov     eax, [ebp+this_object]
mov     edx, [eax]
mov     esi, esp
mov     ecx, [ebp+this_object]
mov     eax, [edx+4]
call    eax
```

Assignment of the vtable to EDX

- The function that will be called will change on runtime

Virtual calls?

```
mov     [ebp+this_object], eax
mov     eax, [ebp+this_object]
mov     edx, [eax]
mov     esi, esp
mov     ecx, [ebp+this_object]
mov     eax, [edx+4]
call    eax
```

Assignment of the vtable to EDX

Move the virtual function to EAX

The virtual call

The Game

String in the Binary

Address	Length	Type	String
.rdata:0000…	00000011	C	Chicken Invaders
.rdata:0000…	0000000D	C	gameOver.ogg
.rdata:0000…	00000020	C	GAME OVER your final score
.rdata:0000…	0000000A	C	claps.ogg
.rdata:0000…	00000026	C	You WON the game\n your final score
.rdata:0000…	00000012	C	

- The string is in use by Controller::runLevel

Controller::runLevel - The SpaceShip

- In the beginning of this function we can see a shared_pointer of a spaceShip object is being used.

```
call    std::shared_ptr<spaceShip>::operator-><spaceShip,0>(void)
```

- We are going to take a deeper look at this object.

“SpaceShip” Constructor

```
mov     rcx, [rbp+1A0h+arg_0] ; this
call    GameObject::GameObject(void)
nop
mov     rax, [rbp+1A0h+arg_0]
lea     rcx, const spaceShip::`vftable'
mov     [rax], rcx
```

“SpaceShip” Constructor

```
mov     rcx, [rbp+1A0h+arg_0] ; this
call    GameObject::GameObject(void)
nop
mov     rax, [rbp+1A0h+arg_0]
lea     rcx, const spaceShip::`vftable'
mov     [rax], rcx
```

“SpaceShip” Constructor

```
mov     rcx, [rbp+1A0h+arg_0] ; this
call    GameObject::GameObject(void)
nop
mov     rax, [rbp+1A0h+arg_0]
lea     rcx, const spaceShip::`vftable'
mov     [rax], rcx
```

“SpaceShip” Vtable

```
                dq offset const spaceShip::`RTTI Complete Object Locator'
const spaceShip::`vftable' dq offset GameObject::draw(sf::RenderWindow &)
                                     ; DATA XREF: spaceShip::spaceShip(voi
                dq offset spaceShip::moveObj(float,direction)
                dq offset GameObject::setPosition(sf::Vector2<float>)
                dq offset GameObject::getLive(void)
                dq offset GameObject::decreaseLives(void)
```

“SpaceShip” Vtable

```
                dq offset const spaceShip::`RTTI Complete Object Locator'
const spaceShip::`vftable' dq offset GameObject::draw(sf::RenderWindow &)
                                     ; DATA XREF: spaceShip::spaceShip(voi
                dq offset spaceShip::moveObj(float,direction)
                dq offset GameObject::setPosition(sf::Vector2<float>)
                dq offset GameObject::getLive(void)
                dq offset GameObject::decreaseLives(void)
```

“SpaceShip” Vtable

```
                dq offset const spaceShip::`RTTI Complete Object Locator'
const spaceShip::`vftable' dq offset GameObject::draw(sf::RenderWindow &)
                                     ; DATA XREF: spaceShip::spaceShip(voi
                dq offset spaceShip::moveObj(float,direction)
                dq offset GameObject::setPosition(sf::Vector2<float>)
                dq offset GameObject::getLive(void)
                dq offset GameObject::decreaseLives(void)
```

- We can see the vtable contains both spaceShip function and GameObject functions
- **SpaceShip Inherits from GameObject**

How does it look like from the father side?

The “Game Object” constructor:: vtable

- The game object assigns its vtable to the first 8 bytes.

```
lea     rcx, const GameObject::`vftable'
mov     [rax], rcx
```

“Game Object” constructor:: members

```
lea     rcx, [rbp+130h+var_6C]
call    cs:sf::VideoMode::getDesktopMode(void)
mov     rcx, [rbp+130h+this]
mov     eax, [rax+4]
mov     [rcx+12], eax
lea     rcx, [rbp+130h+var_4C]
call    cs:sf::VideoMode::getDesktopMode(void)
mov     rcx, [rbp+130h+this]
mov     eax, [rax]
mov     [rcx+8], eax
mov     rax, [rbp+130h+this]
```

“Game Object” constructor:: members

```
lea     rcx, [rbp+130h+var_6C]
call    cs:sf::VideoMode::getDesktopMode(void)
mov     rcx, [rbp+130h+this]
mov     eax, [rax+4]
mov     [rcx+12], eax
lea     rcx, [rbp+130h+var_4C]
call    cs:sf::VideoMode::getDesktopMode(void)
mov     rcx, [rbp+130h+this]
mov     eax, [rax]
mov     [rcx+8], eax
mov     rax, [rbp+130h+this]
```

“Game Object” :: members

```
lea     rcx, [rbp+130h+var_6C]
call    cs:sf::VideoMode::getDesktopMode(void)
mov     rcx, [rbp+130h+this]
mov     eax, [rax+4]
mov     [rcx+12], eax
lea     rcx, [rbp+130h+var_4C]
        eoMode::getDesktopMode(void)
        30h+this]
mov     eax, [rax]
mov     [rcx+8], eax
mov     rax, [rbp+130h+this]
```

Assignment to the relevant offset in the GameObject

“Game Object” :: members

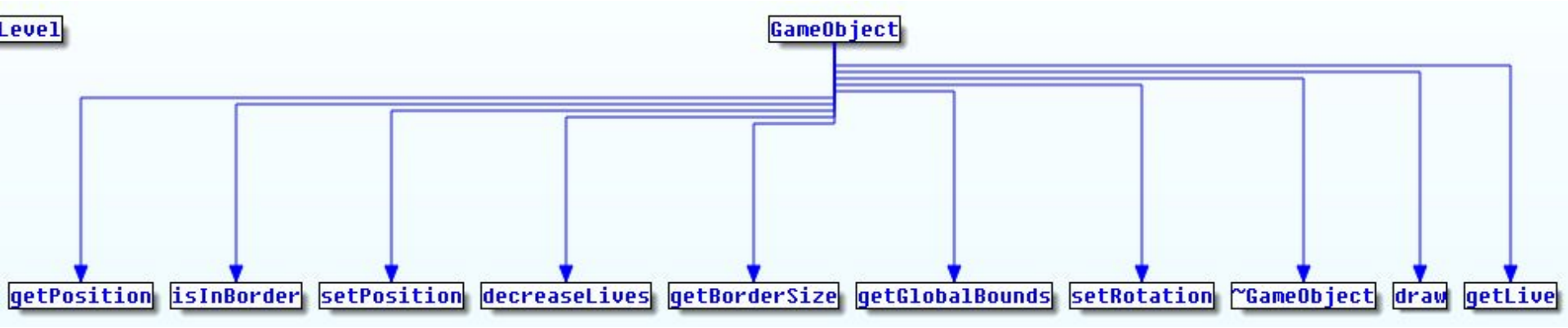
```
lea     rcx, [rbp+130h+var_6C]
call    cs:sf::VideoMode::getDesktopMode(void)
mov     rcx, [rbp+130h+this]
mov     eax, [rax+4]
mov     [rcx+12], eax
lea     rcx, [rbp+130h+var_4C]
mov     eax, [rax]
mov     [rcx+8], eax
mov     rax, [rbp+130h+this]
```

Assignment to the relevant offset int the Game object

The return values from the function calls

“Game Object” :: Vtable

- If we return to our vtable these are the functions of the game object



“Game Object” :: Vtable

```
                dq offset const GameObject::`RTTI Complete Object Locator'
const GameObject::`vftable' dq offset GameObject::draw(sf::RenderWindow &)
                                      ; DATA XREF: GameObject::GameObject(void)
                dq offset j__purecall_0
                dq offset GameObject::setPosition(sf::Vector2<float>)
                dq offset GameObject::getLive(void)
                dq offset GameObject::decreaseLives(void)
                dq 0
```

“Game Object” :: Vtable

```
                dq offset const GameObject::`RTTI Complete Object Locator'
const GameObject::`vftable' dq offset GameObject::draw(sf::RenderWindow &)
                                      ; DATA XREF: GameObject::GameObject(void)
                dq offset j__purecall 0
                dq offset GameObject::setPosition(sf::Vector2<float>)
                dq offset GameObject::getLive(void)
                dq offset GameObject::decreaseLives(void)
                dq 0
```

- We can see the function “spaceShip::moveObj” is a pure virtual function in the father game object.

REcap

```
class GameObject {

public:
  GameObject();
  virtual void moveObj(float, direction) = 0;
  virtual void setPosition(sf::Vector2f pos);
  virtual int getLive() const;
  virtual void decreaseLives();

protected:
  sf::Vector2u windowSize;
  int m_lives;
};
```

```
                dq offset ??_R4GameObject@@6B@ ; const GameObject::RTTI Compie
 const GameObject::`vftable'
?_7GameObject@@6B@ dq offset j_?draw@GameObject@@UEAAXAEAVRenderWindow@sf@@@Z
                                        ; DATA XREF: GameObject::GameObject(voi
                                        ; GameObject::draw(sf::RenderWindow &)
                dq offset j__purecall_0
                dq offset j_?setPosition@GameObject@@UEAAXV?$Vector2@M@sf@@@Z ;
                dq offset j_?getLive@GameObject@@UEBAHXZ ; GameObject::getLive(
                dq offset j_?decreaseLives@GameObject@@UEAAXXZ ; GameObject::de
                db    0
```

REcap

```
class spaceShip : public GameObject
{
public:
  spaceShip();
  ... ?? ..
  virtual void moveObj(float, direction);
  .. ?? ..

};
```

```
; const spaceShip::`vftable'
??_7spaceShip@@6B@ dq offset j_?draw@GameObject@@UEAAXAEAVRenderWindow@sf@@@Z
                                        ; DATA XREF: spaceShip::spaceShip(void)+55↑o
                                        ; GameObject::draw(sf::RenderWindow &)
                dq offset j_?moveObj@spaceShip@@UEAAXMW4direction@@@Z ; spaceShip::moveObj(float,direction)
                dq offset j_?setPosition@GameObject@@UEAAXV?$Vector2@M@sf@@@Z ; GameObject::setPosition(sf::Vector2<float>)
                dq offset j_?getLive@GameObject@@UEBAHXZ ; GameObject::getLive(void)
                dq offset j_?decreaseLives@GameObject@@UEAAXXZ ; GameObject::decreaseLives(void)
```

After understanding those Objects and Inheritance, it’s time to go back to the code flow

Controller::runLevel - assembly

- In the beginning we had the spaceship shared_pointer initialization.
- Afterwards we can find the following function call:

```
mov     rcx, [rbp+3A0h+arg_0] ; this
call    Controller::fillWavesVector(void)
```

- We need to dig deeper into this function and understand what happens there.

fillWavesVector:: Emplace_Pack Wave objects

```
call    std::vector<std::unique_ptr<WaveInterface,std::default_delete<WaveIn
```

fillWavesVector:: Emplace_Pack Wave objects

```
call    std::vector<std::unique_ptr<WaveInterface,std::default_delete<WaveIn
```

```
e<WaveInterface>>>>::emplace_back<std::unique_ptr<Wave2,std::de
```

```
call    std::make_unique<Wave2,,0>(void)
mov     [rbp+1C0h+var_28], rax
mov     rax, [rbp+1C0h+var_28]
mov     [rbp+1C0h+var_20], rax
mov     rax, [rbp+1C0h+arg_0]
add     rax, 680h
mov     rdx, [rbp+1C0h+var_20]
mov     rcx, rax
call    std::vector<std::unique_ptr<WaveInterfa
```

fillWavesVector:: Emplace_Pack Wave objects

- We can understand this function emplaces object in a vector.
- These object are unique pointers that represent every level named “wave”

```
call    std::vector<std::unique_ptr<WaveInterface,std::default_delete<WaveIn
```

```
call    std::make_unique<Wave2,,0>(void)
mov     [rbp+1C0h+var_28], rax
mov     rax, [rbp+1C0h+var_28]
mov     [rbp+1C0h+var_20], rax
mov     rax, [rbp+1C0h+arg_0]
add     rax, 680h
mov     rdx, [rbp+1C0h+var_20]
mov     rcx, rax
call    std::vector<std::unique_ptr<WaveInterfa
```

fillWavesVector:: Emplace_back WaveBoss object

- When we look further we realize that the function emplaces more unique pointers of objects like WaveBosses. Uhm not suspicious at all!

```
call    std::make_unique<WaveBoss2,,0>(void)
mov     [rbp+1C0h+var_28], rax
mov     rax, [rbp+1C0h+var_28]
mov     [rbp+1C0h+var_20], rax
mov     rax, [rbp+1C0h+arg_0]
add     rax, 680h
mov     rdx, [rbp+1C0h+var_20]
mov     rcx, rax
call    std::vector<std::unique_ptr<WaveInterface,std::default_c
nop
```

Controller::fillWavesVector Code

```
void Controller::fillWavesVector()
{


  m_waves.emplace_back(std::make_unique <Wave1>());
  m_waves.emplace_back(std::make_unique <Wave2>());
  m_waves.emplace_back(std::make_unique <Wave3>());
  m_waves.emplace_back(std::make_unique <Wave4>());
  m_waves.emplace_back(std::make_unique <WaveBoss1>());
  m_waves.emplace_back(std::make_unique <WaveBoss3>());
  m_waves.emplace_back(std::make_unique <WaveBoss2>());
}
```

Controller::runLevel Using the Wave vector

- Before delving into understanding the waveBoss2 let’s check the usage of the wave’s vector.

Controller::runLevel Using the Wave vector

- Before delving into understanding the waveBoss2 let's check the usage of the wave's vector.
-

```
mov     [rbp+3A0h+counter], 0
mov     rcx, [rbp+3A0h+p_vector] ; this
call    Controller::fillWavesVector(void)
```

Controller::runLevel Using the Wave vector

```
mov     rcx, [rbp+3A0h+counter]
call    std::vector<std::unique_ptr<WaveInterface,std::default_delete
```

```
.ete<WaveInterface>>>>::operator[](unsigned __int64)
```

```
call    std::unique_ptr<WaveInterface,std::default_delete<Wav
mov     [rbp+3A0h+p_wave], rax
```

Controller::runLevel- Virtual Call

- There are lots problem for reverse engineers when virtual call is been used.
- We can not easily know statically know which function is going to be called.

```
mov     rcx, [rbp+3A0h+p_wave]
call    qword ptr [rax]
```

REcap

- After reading the assembly code, so far we figure out some of the parts!

```
void Controller::runLevel()
{
  m_player->restart();
  unsigned waveNumber = 0;
  fillWavesVector();

  do {
    m_gameObjectVector.emplace_back(m_player);
    m_player->resetPosition();
    m_window.clear();
    moveBetweenWaves(waveNumber + 1);
    m_waves[waveNumber]->createWave();
    m_waves[waveNumber]->getWaveStage(m_gameObjectVector);
    m_status = m_level.runNewLevel(waveNumber);
    waveNumber++;
    m_gameObjectVector.resize(0);
  } while (m_status == CONTINUE_T && waveNumber != m_waves.size());

  m_status == GAME_OVER_T ? gameEnded("     GAME OVER your final score", "gameOver.ogg") :
                            gameEnded(" You WON the game\n   your final score", "claps.ogg");

  m_gameObjectVector.resize(0);
}
```

Examine the virtual call on runtime

```
.text:00007FF6C34D75E2 mov     edx, eax
.text:00007FF6C34D75E4 call    std::vector<std::unique_ptr<WaveInterface,std::de
.text:00007FF6C34D75E9 mov     rcx, rax
.text:00007FF6C34D75EC call    std::unique_ptr<WaveInterface,std::default_delete
.text:00007FF6C34D75F1 mov     [rbp+3A0h+p_wave], rax
.text:00007FF6C34D75F8 mov     rax, [rbp+3A0h+p_wave]
.text:00007FF6C34D75FF mov     rax, [rax]
.text:00007FF6C34D7602 mov     rcx, [rbp+3A0h+p_wave]
.text:00007FF6C34D7609 call    qword ptr [rax]
.text:00007FF6C34D760B mov     rax, [rbp+3A0h+arg_0]
.text:00007FF6C34D7612 add     rax, 8
.text:00007FF6C34D7616 mov     [rbp+3A0h+p_wave], rax
.text:00007FF6C34D761D mov     ecx, [rbp+3A0h+var_39C]
.text:00007FF6C34D7620 mov     rdx, [rbp+3A0h+arg_0]
.text:00007FF6C34D7627 add     rdx, 680h
.text:00007FF6C34D762E mov     [rbp+3A0h+var_40], rdx
.text:00007FF6C34D7635 mov     edx, ecx
.text:00007FF6C34D7637 mov     rcx, [rbp+3A0h+var_40]
```

```
RAX 00007FF6C3559EA8
RBX 0000000000000000
RCX 00000205EA243B20
RDX 0000000000000002
RSI 0000000000000000
RDI 000000BADA5AEFC8
RBP 000000BADA5AEC30
RSP 000000BADA5AEC00
RIP 00007FF6C34D7609
R8  00000205DEE91BC0
R9  0000000000008000
R10 00000000FFFFFFFF
R11 00000205EEFDFA10
R12 0000000000000000
R13 0000000000000000
R14 0000000000000000
R15 0000000000000000
EFL 00000202
```

Examine the virtual call on runtime

```
.text:00007FF6C34D75E2 mov     edx, eax
.text:00007FF6C34D75E4 call    std::vector<std::unique_ptr<WaveInterface,std::de
.text:00007FF6C34D75E9 mov     rcx, rax
.text:00007FF6C34D75EC call    std::unique_ptr<WaveInterface,std::default_delete
.text:00007FF6C34D75F1 mov     [rbp+3A0h+p_wave], rax
.text:00007FF6C34D75F8 mov     rax, [rbp+3A0h+p_wave]
.text:00007FF6C34D75FF mov     rax, [rax]
.text:00007FF6C34D7602 mov     rcx, [rbp+3A0h+p_wave]
.text:00007FF6C34D7609 call    qword ptr [rax]
.text:00007FF6C34D760B mov     rax, [rbp+3A0h+arg_0]
.text:00007FF6C34D7612 add     rax, 8
.text:00007FF6C34D7616 mov     [rbp+3A0h+p_wave], rax
.text:00007FF6C34D761D mov     ecx, [rbp+3A0h+var_39C]
.text:00007FF6C34D7620 mov     rdx, [rbp+3A0h+arg_0]
.text:00007FF6C34D7627 add     rdx, 680h
.text:00007FF6C34D762E mov     [rbp+3A0h+var_40], rdx
.text:00007FF6C34D7635 mov     edx, ecx
.text:00007FF6C34D7637 mov     rcx, [rbp+3A0h+var_40]
```

```
RAX 00007FF6C3559EA8
RCX 00000205EA243B20
RDX 0000000000000002
RSI 0000000000000000
RDI 000000BADA5AEFC8
RBP 000000BADA5AEC30
RSP 000000BADA5AEC00
RIP 00007FF6C34D7609
R8  00000205DEE91BC0
R9  0000000000008000
R10 00000000FFFFFFFF
R11 00000205EEFDFA10
R12 0000000000000000
R13 0000000000000000
R14 0000000000000000
R15 0000000000000000
EFL 00000202
```

Examine the virtual call on runtime

```
.text:00007FF6C34D75E2 mov     edx, eax
.text:00007FF6C34D75E4 call    std::vector<std::unique_ptr<WaveInterface,std::de
.text:00007FF6C34D75E9 mov     rcx, rax
.text:00007FF6C34D75EC call    std::unique_ptr<WaveInterface,std::default_delete
.text:00007FF6C34D75F1 mov     [rbp+3A0h+p_wave], rax
.text:00007FF6C34D75F8 mov     rax, [rbp+3A0h+p_wave]
.text:00007FF6C34D75FF mov     rax, [rax]
.text:00007FF6C34D7602 mov     rcx, [rbp+3A0h+p_wave]
.text:00007FF6C34D7609 call    qword ptr [rax]
.text:00007FF6C34D760B mov     rax, [rbp+3A0h+arg_0]
.text:00007FF6C34D7612 add     rax, 8
.text:00007FF6C34D7616 mov     [rbp+3A0h+p_wave], rax
.text:00007FF6C34D761D mov     ecx, [rbp+3A0h+var_39C]
.text:00007FF6C34D7620 mov     rdx, [rbp+3A0h+arg_0]
.text:00007FF6C34D7627 add     rdx, 680h
.text:00007FF6C34D762E mov     [rbp+3A0h+var_40], rdx
.text:00007FF6C34D7635 mov     edx, ecx
.text:00007FF6C34D7637 mov     rcx, [rbp+3A0h+var_40]
```

```
RAX 00007FF6C3559EA8
RBX 0000000000000000
RCX 00000205EA243B20
RDX 0000000000000002
RSI 0000000000000000
RDI 000000BADA5AEFC8
RBP 000000BADA5AEC30
RSP 000000BADA5AEC00
RIP 00007FF6C34D7609
R8  00000205DEE91BC0
R9  0000000000008000
```

```
RAX 00007FF6C3559EA8
RBX 0000000000000000
RCX 00000205EA243B20
RDX 0000000000000002
RSI 0000000000000000
RDI 000000BADA5AEFC8
RBP 000000BADA5AEC30
RSP 000000BADA5AEC00
RIP 00007FF6C34D7609
R8  00000205DEE91BC0
R9  0000000000008000
R10 00000000FFFFFFFF
R11 00000205EEFDFA10
R12 0000000000000000
R13 0000000000000000
R14 0000000000000000
R15 0000000000000000
EFL 00000202
```

Virtual call to Wave3::createWave()

- The address of the the register EAX(contains the function address of the virtual call) is Wave3::createWave()

```
.rdata:00007FF6C3559EA0 dq offset const Wave3::`RTTI Complete Object Locator'
.rdata:00007FF6C3559EA8 const Wave3::`vftable' dq offset Wave3::createWave(void)
.rdata:00007FF6C3559EA8                                                 ; DATA XREF: Wave3
```

Examine the virtual call on runtime

```
00007FF6C34D75E2 mov     edx, eax
00007FF6C34D75E4 call    std::vector<std::unique_ptr<Wave]
00007FF6C34D75E9 mov     rcx, rax
00007FF6C34D75EC call    std::unique_ptr<WaveInterface,std
00007FF6C34D75F1 mov     [rbp+3A0h+p_wave], rax
00007FF6C34D75F8 mov     rax, [rbp+3A0h+p_wave]
00007FF6C34D75FF mov     rax, [rax]
00007FF6C34D7602 mov     rcx, [rbp+3A0h+p_wave]
00007FF6C34D7609 call    qword ptr [rax]
00007FF6C34D760B mov     rax, [rbp+3A0h+arg_0]
00007FF6C34D7612 add     rax, 8
00007FF6C34D7616 mov     [rbp+3A0h+p_wave], rax
00007FF6C34D761D mov     ecx, [rbp+3A0h+var_39C]
00007FF6C34D7620 mov     rdx, [rbp+3A0h+arg_0]
00007FF6C34D7627 add     rdx, 680h
```

```
RAX 00007FF6C3559EC0 ↳ .rdata:const Wave4::`vftable'
RCX 00000205EA243A40 ↳ debug183:00000205EA243A40
RDX 0000000000000003 ↳
RSI 0000000000000000 ↳
RDI 000000BADA5AEFC8 ↳ Stack[00002EC8]:000000BADA5AEFC8
RBP 000000BADA5AEC30 ↳ Stack[00002EC8]:000000BADA5AEC30
RSP 000000BADA5AEC00 ↳ Stack[00002EC8]:000000BADA5AEC00
RIP 00007FF6C34D7609 ↳ Controller::runLevel(void)+129
R8  00000205DEE91BC0 ↳ debug054:00000205DEE91BC0
R9  0000000000008000 ↳
R10 00000000FFFFFFFF ↳
R11 00000205EEFE0000 ↳ debug206:00000205EEFE0000
R12 0000000000000000 ↳
R13 0000000000000000 ↳
R14 0000000000000000 ↳
R15 0000000000000000 ↳
EFL 00000202
```

Examine the virtual call on runtime

```
00007FF6C34D75E2 mov     edx, eax
00007FF6C34D75E4 call    std::vector<std::unique_ptr<Wave
00007FF6C34D75E9 mov     rcx, rax
00007FF6C34D75EC call    std::unique_ptr<WaveInterface,st
00007FF6C34D75F1 mov     [rbp+3A0h+p_wave], rax
00007FF6C34D75F8 mov     rax, [rbp+3A0h+p_wa
00007FF6C34D75FF mov     rax, [rax]
00007FF6C34D7602 mov     rcx, [rbp+3A0h+p_wav
00007FF6C34D7609 call    qword ptr [rax]
00007FF6C34D760B mov     rax, [rbp+3A0h+arg_0
00007FF6C34D7612 add     rax, 8
00007FF6C34D7616 mov     [rbp+3A0h+p_wave], r
00007FF6C34D761D mov     ecx, [rbp+3A0h+var_3
00007FF6C34D7620 mov     rdx, [rbp+3A0h+arg_0]
00007FF6C34D7627 add     rdx, 680h
```

```
RAX 00007FF6C3559EC0 ↳ .rdata:const Wave4::`vftable'
RCX 00000205EA243A40 ↳ debug183:00000205EA243A40
RDX 0000000000000003 ↳
RSI 0000000000000000 ↳
RDI 000000BADA5AEFC8 ↳ Stack[00002EC8]:000000BADA5AEFC8
RBP 000000BADA5AEC30 ↳ Stack[00002EC8]:000000BADA5AEC30
RSP 000000BADA5AEC00 ↳ Stack[00002EC8]:000000BADA5AEC00
RIP 00007FF6C34D7609 ↳ Controller::runLevel(void)+129
R8  00000205DEE91BC0 ↳ debug054:00000205DEE91BC0
R9  0000000000008000 ↳
R10 00000000FFFFFFFF ↳
R11 00000205EEFE0000 ↳ debug206:00000205EEFE0000
```

```
RAX 00007FF6C3559EC0 ↳
RBX 0000000000000000 ↳
RCX 00000205EA243A40 ↳
RDX 0000000000000003 ↳
RSI 0000000000000000 ↳
RDI 000000BADA5AEFC8 ↳
```

Virtual call to Wave4::createWave()

- The next time we ran the code, EAX points to Wave4::createWave()

```
.rdata:00007FF6C3559EB8 dq offset const Wave4::`RTTI Complete Object Locator'
.rdata:00007FF6C3559EC0 const Wave4::`vftable' dq offset Wave4::createWave(void)
.rdata:00007FF6C3559EC0                                          ; DATA XREF: Wave4
```

The virtual call - createWave()

- After running the code multiple times, we discovered the virtual call is to createWave() of the relevant Wave object.
- All the wave objects are part of the vector explained before and being chosen according to a counter

The virtual call - createWave()

- After running the code multiple times, we discovered the virtual call is to *createWave* of the relevant Wave object.
- All the wave objects are part of the vector explained before and beeing chose according to a counter
- When a boss appear in the game the "*WaveBossX::createWave*" function is being called.

```
00007FF6C3559E58 dq offset const WaveBoss2::`RTTI Complete Object Locator'
00007FF6C3559E60 const WaveBoss2::`vftable' dq offset WaveBoss2::createWave(void)
00007FF6C3559E60                                      ; DATA XREF: WaveBoss2::W
```


Score: 1950
Lives: 1
Examine WaveBoss2
get background at www.twitterevolutions.com

WaveBoss2 constructor

```
lea     rcx, const ChickenBoss2::`vftable'
mov     [rax], rcx
mov     rax, [rbp+1D0h+arg_0]
mov     dword ptr [rax+18h], 15
```

- “WaveBoss” Inherits from ChickenBoss2

WaveBoss2 constructor

```
lea     rcx, const ChickenBoss2::`vftable'
mov     [rax], rcx
mov     rax, [rbp+1D0h+arg_0]
mov     dword ptr [rax+18h], 15
```

The Flow from here

- We can see that “WaveBoss2” Inherits from “ChickenBoss2”
- Our next step is to reverse engineer the code of ChickenBoss2 and figure out what we would want to change in the boss.

Summary

- It is quite similar to the process we've done so far I'll summarize the results.

Summary

- It is quite similar to the process we've done so far I'll summarize the results.
- After reversing the "ChickenBoss2" object we figure out it also inherits from "ChickenBase" that inherits from "GameObject"

WaveBoss2 → ChickenBoss2 → ChickenBase → GameObject

Summary

- It is quite similar to the process we've done so far I'll summarize the results.
- After reversing the "ChickenBoss2" object we figure out it also inherits from "ChickenBase" that inherits from "GameObject"
- We also saw "GameObject" has other children like: regularChicken, spaceShip, etc.
    - After reversing GameObject we saw one of its members seems like the amount of lives the object has.

Summary

- It is quite similar to the process we've done so far I'll summarize the results.
- After reversing the "ChickenBoss2" object we figure out it also inherits from "ChickenBase" that inherits from "GameObject"
- We also saw "GameObject" has other children like: regularChicken, spaceShip, etc.
  - After reversing GameObject we saw one of its members seems like the amount of lives the object has.
  - Now we know what to change!

Changing the boss lives

- After the research we've done now we can understand what is this member:

```
lea     rcx, const ChickenBoss2::`vftable'
mov     [rax], rcx
mov     rax, [rbp+1D0h+arg_0]
mov     dword ptr [rax+18h], 15
```

- The amount of lives the boss has

Changing the boss lives

```
E8 4D 06 FE FF                call    ChickenBase::ChickenBase(void)
90                            nop
48 8B 85 E0 01 00 00          mov     rax, [rbp+1D0h+arg_0]
48 8D 0D 24 57 0A 00          lea     rcx, const ChickenBoss2::`vftable'
48 89 08                      mov     [rax], rcx
48 8B 85 E0 01 00 00          mov     rax, [rbp+1D0h+arg_0]
C7 40 18 0F 00 00 00          mov     dword ptr [rax+18h], 15
```

Changing the boss lives

```
E8 4D 06 FE FF                  call    ChickenBase::ChickenBase(void)
90                              nop
48 8B 85 E0 01 00 00            mov     rax, [rbp+1D0h+arg_0]
48 8D 0D 24 57 0A 00            lea     rcx, const ChickenBoss2::`vftable'
48 89 08                        mov     [rax], rcx
48 8B 85 E0 01 00 00            mov     rax, [rbp+1D0h+arg_0]
C7 40 18 0F 00 00 00            mov     dword ptr [rax+18h], 15
```

Changing the boss lives

```
E8 4D 06 FE FF                  call    ChickenBase::ChickenBase(void)
90                              nop
48 8B 85 E0 01 00 00            mov     rax, [rbp+1D0h+arg_0]
48 8D 0D 24 57 0A 00            lea     rcx, const ChickenBoss2::`vftable'
48 89 08                        mov     [rax], rcx
48 8B 85 E0 01 00 00            mov     rax, [rbp+1D0h+arg_0]
C7 40 18 0F 00 00 00            mov     dword ptr [rax+18h], 15
```

Changing the boss lives

```
E8 4D 06 FE FF                  call    ChickenBase::ChickenBase(void)
90                              nop
48 8B 85 E0 01 00 00            mov     rax, [rbp+1D0h+arg_0]
48 8D 0D 24 57 0A 00            lea     rcx, const ChickenBoss2::`vftable'
48 89 08                        mov     [rax], rcx
48 8B 85 E0 01 00 00            mov     rax, [rbp+1D0h+arg_0]
C7 40 18 01 00 00 00            mov     dword ptr [rax+18h], 1
```

Questions?

@0xgalz

Thank you for your time

@0xgalz