UNIVERSIDADE FEDERAL DA BAHIA

ANÁLISE DE CÓDIGO ASSEMBLY GERADO POR DESASSEMBLADOR DE SOFTWARES

Trabalho solicitado pelo professor Roberto Figueiredo, como parte da avaliação para a disciplina de Programação de Software Básico.

JOÃO PEDRO RODRIGUES CERQUEIRA
E
ROBSON SANTOS SOUSA

Salvador - BA
28 de Outubro de 2016

Introdução

O Trabalho desenvolvido é um estudo da Engenharia Reversa de um jogo de xadrez para terminal linux. O trabalho foi de identificação das funcionalidades do jogo através do código assembly gerada pelo processo de reversão.

Salvador - BA
28 de Outubro de 2016

Material utilizado

- IDA PRO Free- Disassembler;
    - Versão: 5.0
- ATOM - Editor de texto;
    - Versão: 1.10.2
- Linux Mint - Sistema Operacional Open Source
    - Distributor ID: LinuxMint
    - Description: Linux Mint 18 Sarah
    - Release: 18
    - Codename: sarah
- GitHub - Gerenciador de código versionado com Git
- Git - Sistema de controle de versão
    - Versão: 2.7.4
- Livro: Practical Malware Analysis
    - Autores: Michael Sikorski e Andrew Honing
- Google Docs: Sistema de criação e gerenciamento de texto compartilhado e em tempo real.

GitHub

Este projeto está disponível aqui.

Salvador - BA
28 de Outubro de 2016

Funcionamento do Programa:

O programa escolhido foi um jogo de Xadrez para dois jogadores.

O programa utiliza próprio terminal para exibir o tabuleiro e as peças. Ambos representados alinhamento de caracteres.

O cursor é representado por uma sequência de @'s, seu movimento é realizado através das teclas **'a'** (esquerda), **'s'** (baixo), **'w'** (cima) e **'d'** (direita), toda em caixa baixa. Cada movimento deve ser confirmado com a tecla 'Enter', e só pode ser executado 1 (um) movimento por interação.
Posicionando o cursor na peça escolhida, o usuário deve teclar **'f'** para selecioná-la. Selecionando, o jogador deve escolher o movimento que deseja realizar, posicionando o cursor até a posição de destino, teclando **'f'** para confirmar.
Os movimentos devem respeitar as regras originais do xadrez.

Logo em seguida é a vez do segundo jogador movimentar uma peça.
O processo é idêntico ao do jogador anterior, com o detalhe de que suas peças ficam na parte superior do tabuleiro.

Os jogadores alternam seus movimentos até ocorrer um Xeque-mate que, no jogo de xadrez, nada mais é do que a derrota do Rei.

Após o Xeque-mate o jogo é reiniciado, e as peças são reposicionadas.

Salvador - BA
28 de Outubro de 2016

Desassemblagem

Por não termos sistema operacional Windows para trabalhar, optamos por desassemblar um executável para linux, visto que a aplicação escolhida não tinha características de interface gráfica, e boa portabilidade.

IDA Pro:

IDA PRO é um disassembler muito poderoso distribuído pela Hex Rays. Com ele é possível analisar os mais variados malwares, trabalhar com engenharia reversa e análise de vulnerabilidade.

Desassemblagem - IDA

Escolhemos o IDA pro porque é o software de versão gratuita disponível mais completo, e de fácil utilização.

Usamos o **modo gráfico** para nos orientar acerca da lógica da aplicação, tal como a sua estrutura. Além disso, as funcionalidades **janelas de funções, IDA View-A e Strings** foram muito úteis para nossa análise com seus seus elementos visuais.

Do **modo texto**, exportamos o código assembler (.asm), estudado e comentado posteriormente.

Análise Estática

Análise Estática descreve o processo de analisar o código ou estrutura de uma programa para determinar sua função.

Com o arquivo assembly gerado, analisamos o código e sua estrutura. Deste destacamos:

Arquitetura:

O programa foi desenvolvido e compilado para ser executado em arquitetura x86 (ou x64 com suporte para x86).

Strings:

Identificamos todas as strings usadas pelo desenvolvedor para interagir com jogadores. Dentre elas, destacamos:

"Get input (aswd for direction, f for click)": sinaliza que o jogador deve usar **'a'**, **'s'**, **'w'** ou **'d'** para mover o cursor pelo tabuleiro, e que **'f'** seleciona a posição;

"**a(knight) s(bishop) d(rook) w(queen)**" : Quando um jogador consegue colocar um Peão na última linha do lado oposto do tabuleiro, esta frase aparece para sinalizar que uma Rainha, um Bispo, um Cavalo ou uma Torre podem ser escolhidos para substituir aquele Peão.
"*** Game Over Player %c win ***" : Sinaliza qual jogador ganhou;
"**cannot recognise the input**": Sinaliza que a entrada é inválida;
"**@@@@@**" : Utilizado para desenhar o seletor
"**current_pos**" : Sinaliza a posição corrente.

Todas estas cadeias estão no seguimento comentado como:
"######## VARIÁVEIS GLOBAIS E MENSAGENS UTILIZADAS NO JOGO"

Funções internas:

Identificamos as funções que compõem programa programa. Estas funçõe podem ser encontradas no seguimento comentado como:

"######## FIM FUNÇÕES INTERNAS UTILIZADAS NO JOGO",
são elas:

- ***initialise:*** inicializa o tabuleiro, as peças e os jogadores;

- ***is_game_over:*** verifica se jogo acabou;
- ***is_in_check:*** verifica se o jogador corrente está em check;
- ***actual_move:*** verifica se o movimento é válido
- ***legal_move:*** verifica se o movimento escolhido para peça é válido;
- ***is_path_clear:*** verifica se o caminho da peça a ser percorrido pela peça selecionada está livre;
- ***promote_pawn:*** promove um peão. O usuário seleciona qual em peça o tornará;
- ***display:*** imprime o tabuleiro com a disposição das peças;
- *manage_input:* gerencia os comandos de entrada via teclado para movimentar e selecionar as peças;
- ***to_pos:*** calcula o identificador único de cada casa do tabuleiro;
- ***process:*** movimenta a peça
- ***is_own_piece:*** verifica se a peça selecionada é o jogador corrente;
- ***game_over*****:** exibe o jogador que ganhou;
- ***main:*** função principal do programa. Dentro dela é chamada às funções **initialise**, **display** e ***manage_input***, process e ***game_over***;

Variáveis globais:

Identificamos as variáveis globais utilizadas pelo programa. Todas se encontram na segmento comentado como

"############ SEGMENTO DE VARIÁVEIS GLOBAIS ##############",
são elas:

- ***selected_pos:*** posição da peça se ele estiver selecionada;
- ***has_selected:*** peça selecionada;
- ***cur_player:*** armazena a posição do jogador, 0 para o primeiro e 1 para o segundo;
- ***castle_flag:*** informa se a posição de rook está válida;
- ***are_marked:*** informa se a peça selecionada está apta para movimento válido;
- ***current_pos:*** posição corrente do cursor;
- ***cur_player:*** jogador atual, 0 para o primeiro e 1 para o segundo;
- ***en_passant_flag:*** informa a linha do peão para mover duas casas;
- ***is_clicked:*** informa que é o usuário que está fazendo iteração

Além dessas, há outras duas variáveis globais, **cells_side** e **cells_type**;

Funções externas:

Identificamos o uso de funções externas, e a referência para bibliotecas padrão da linguagem **C**. Todas se encontram no segmento identificado como:

"########## SEGMENTO DE FUNÇÕES EXTERNAS ##############"

São elas:

- ***stdio.h:*** este cabeçalho define vários macros, três tipos de variáveis e diversas funções de entrada e saída.
    - printf: envia a saída formatada para 'stdout';
    - fwrite: escreve dados na no array passado como parâmetro;
    - puts: grava string em na saída padrão;
    - putchar: grava um caractere em stdout
- ***string.h:*** este cabeçalho define um tipo variável, um macro, e diversas funções para a manipulação de strings.
    - memset Copies the character c (an unsigned char) to the first n characters of the string pointed to, by the argument *str*.
- ***stdlib.h:*** este cabeçalho define quatro tipos de variáveis, várias macros e diversas funções gerais.
    - malloc Allocates the requested memory and returns a pointer to it.
- ***assert.h:*** este cabeçalho fornece acesso ao macro 'assert'.
    - __assert_fail: abort the program after false assertion

Conclusão:

Com a experiência que tivemos, deu para perceber que quanto maior for a complexidade do programa a ser estudado, maior será o código em Assembly gerado, e por consequência, o processo de identificação do código e de seus detalhes será mais trabalhoso.

A análise estática é um processo que exige do programador muito conhecimento sobre a sintaxe do Assembly, e que saiba identificar o que é externo ao código Assembly, e isso se torna mais simples se o programador conhece a linguagem em que o programa foi escrito originalmente. No nosso caso, identificamos o nome de muitas funções utilizadas na linguagem C.

Todo esse processo em larga escala e realizado por profissionais especializados, é de grande importância para as fábricas de software, por exemplo, pois podem tornar seu produto mais competitivo estudando como o software do concorrente trabalha.

Salvador - BA
28 de Outubro de 2016