Engenharia reversa utilizando o x64dbg

Juan Felipe Serafim dos Santos

[1] Centro de Informática
Universidade Federal de Pernambuco (UFPE) – Recife, PE – Brazil

jfss@cin.ufpe.br

Resumo. *Engenharia reversa, no contexto da computação, é o processo de analisar um software, hardware ou sistema digital já existente com o objetivo de compreender seu funcionamento interno, especialmente quando o código-fonte ou a documentação original não estão disponíveis. Em análise de aplicativos compilados, a engenharia reversa pode envolver a descompilação de um executável para que se possa estudar seu código em nível de máquina ou em uma linguagem mais próxima da original, permitindo compreender como certas funcionalidades foram implementadas. Também é comum a utilização de ferramentas de debugging para acompanhar a execução do programa passo a passo e observar o comportamento de variáveis, chamadas de sistema e interações com arquivos ou a rede.*

1. Objetivo

Nessa atividade, será utilizado o software x64dbg. O x64dbg é um *debugger* para *Windows*, no qual verifica, a nível de linguagem de máquina, o funcionamento de uma determinada aplicação. Para tal, iremos utilizar uma aplicação encontrada no site crackmes.one. O Crackmes é uma plataforma para entusiastas treinarem suas habilidades em engenharia reversa, resolvendo problemas e compartilhando suas soluções.

2. Como a ferramenta funciona

Existem vários campos úteis na interface da aplicação. Uma visão geral dela encontra-se abaixo:

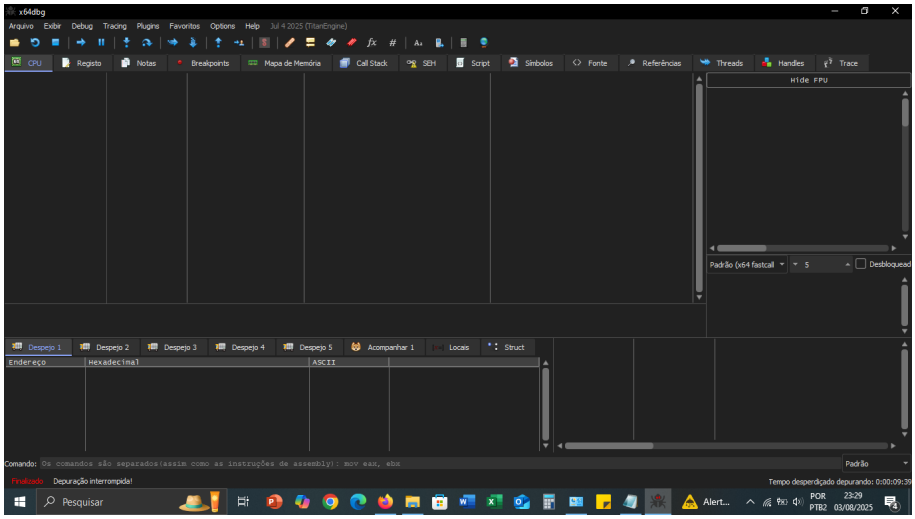

Figura 1. Interface do x64dbg

Para a resolução do problema encontrado no crackmes, iremos utilizar o campo principal do x64dbg, que apresentam todas as instruções do executável. Pela proposta desse problema, serão nessas instruções que estará a solução.

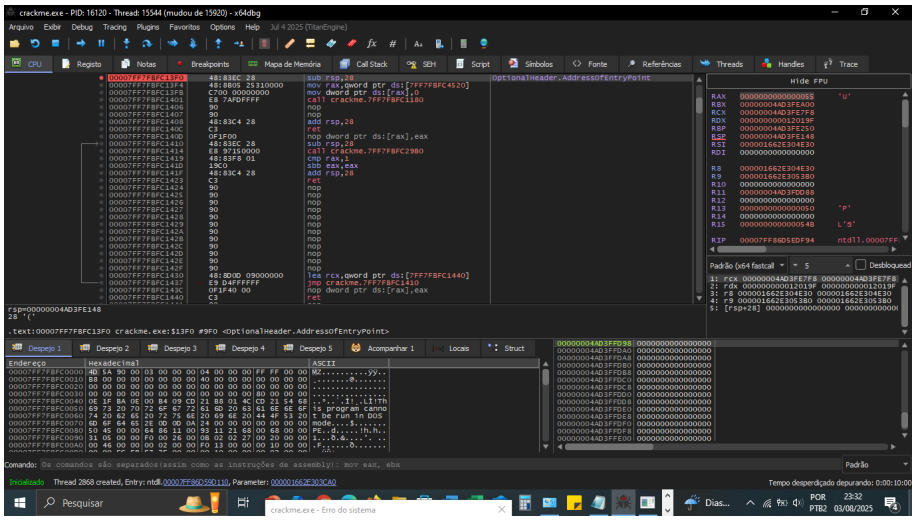

Figura 2. Depuração

Ao percorrer pelas instruções, é possível observar os endereços de tratamento de condicionais do problema, como caso a senha inserida seja incorreta ou proteções contra o uso de debbuger.

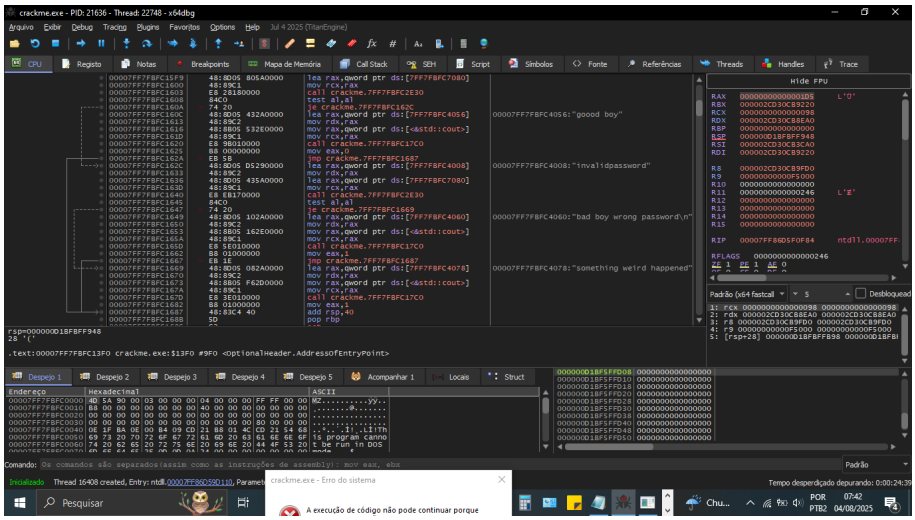

Figura 3. Lista de instruções do app

Como a proposta desse problema é inserir a senha correta, e, através da descrição do problema, foi mencionado que a senha estaria codificada em base64, percebe-se em uma das linhas do executável, uma string referente a potencial senha.

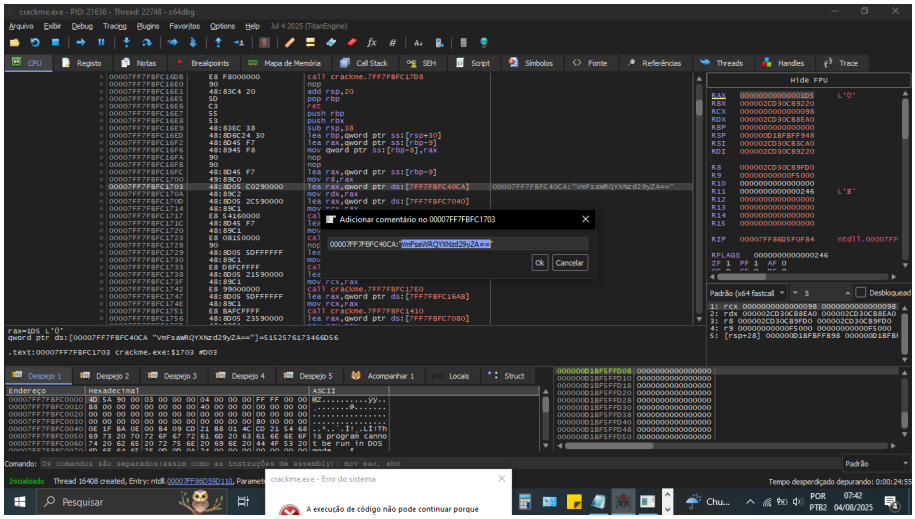

Figura 4. Linha referente a senha

Pelo fato da aplicação ter proteção contra o uso de debbuger, a execução foi realizada de maneira independente, verificando a condição de senha incorreta e senha correta.

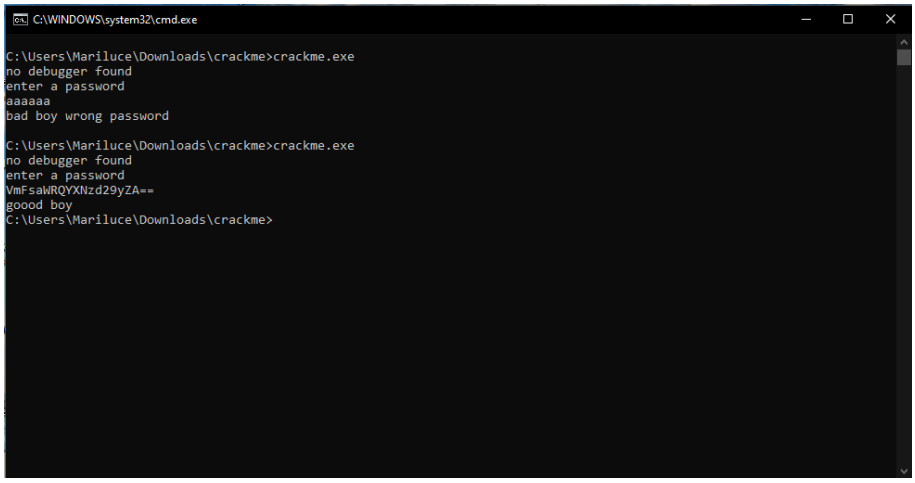

Figura 5. Inserção de senha

3. Análise da vulnerabilidade

Depuração de aplicativo compilado é sempre uma possibilidade. A compreensão de como ele é executado pelas plataformas destino descreve como, em baixo nível, como a aplicação funciona. As possibilidades são diversas, uma vez que passa a ser factível manipular o comportamento do executável em tempo de execução e, por conta disso, burlar medidas de segurança da própria aplicação.

4. Sugestão de defesa

Mesmo que seja complicado realizar a proteção da aplicação contra os *debuggers*, existem medidas para mitigar tais práticas. Uma das técnicas mais comuns é a obfuscação de código, que transforma o código-fonte ou o executável de forma que continue funcional,

mas se torne extremamente difícil de compreender. Além disso, é possível implementar técnicas *anti-debugging*, que consistem em mecanismos para detectar ou dificultar o uso de depuradores, como verificações da presença de ferramentas como o x64dbg ou alterações no comportamento do programa quando monitorado.

Referências

Engenharia reversa: Primeiro contato. https://dev.to/ryan_gozlyngg/engenharia-reversa-primeiro-contato-parte-1-2gih.

'Lilsan44444' (2025). Lilsan44444's validator. https://crackmes.one/crackme/682113c96297cca3ff7d7834.

Ogilvie, D. (2014). x64dbg - an open-source x64/x32 debugger for windows. https://x64dbg.com/.