Fichamento e Análise Comparativa entre “Assembly Reverse Analysis on Malicious Code of Web Rootkit Trojan”[1] e “Opcode and API Based Machine Learning Framework For Malware Classification”[2]

Fernando Antonio M. Schettini[1]

[1]Universidade Federal da Bahia
40170-115 – Salvador – BA – Brasil

fernandoschettini@Outlook.com

Abstract. *This document is a summary of two scientific articles, discussing their objectives, tools used, results and conclusions, carried out as an academic activity of the Basic Software Programming (MATA49) course taught by Professor Mirlei Moura da Silva in the first semester of 2023. The main focus of the articles is the use of Assembly language at the level of basic software to solve proposed problems, and a comparative analysis of the two articles is also made, highlighting their similarities and differences.*

Resumo. Este documento é um fichamento de dois artigos científicos, abordando seus objetivos, ferramentas utilizadas, resultados e conclusões realizado como atividade acadêmica da disciplina de Programação de Software Básico (MATA49) lecionada pela professora Mirlei Moura da Silva, no primeiro semestre de 2023. O foco principal dos artigos é a utilização da linguagem Assembly a nível de software básico para resolução de problemas propostos, aqui também é feito uma análise comparativa dos dois artigos, destacando suas semelhanças e diferenças.

1. Fichamento do artigo “Assembly Reverse Analysis on Malicious Code of Web Rootkit Trojan”

O artigo “*Assembly Reverse Analysis on Malicious Code of Web Rootkit Trojan*", foi publicado em 2009 na *International Conference on Web Information Systems and Mining*, que aconteceu em Shanghai na China pelos autores Yong Wang, Dawu Gu, Jianping Xu e Fenyu Zen.

1.1 Objetivos

O principal objetivo do artigo é o processo de análise reversa de dois trojans famosos: *machine dog Trojan* e *BIOS Trojan*. Desta forma, é possível descobrir como funcionam e, consequentemente, descobrir formas de se proteger desses trojans.

Como parte desse objetivo maior, foi-se realizado uma análise inicial de dois outros códigos maliciosos, *CIH Vírus* e um *rootkit Trojan*, utilizando de dois assemblers diferentes.

Logo depois, a análise reversa foi feita utilizando várias ferramentas e obtendo várias informações do vírus.

1.2 Ferramentas Utilizadas

- Turbo Assembler (TASM):

"Turbo Assembler (TASM) é um montador da Borland Turbo. O Turbo Assembler 5.0 é um montador completo e autônomo que inclui muitas ferramentas necessárias para criar e depurar programas de linguagem assembly para plataformas DOS e Windows de 16 e 32 bits, incluindo Windows 3.X, Win95, Win98 e NT. Algumas das ferramentas incluídas são montadores, linkers, depuradores estilo console e compiladores de recursos. Cada uma dessas ferramentas vem em uma versão de 16 bits e 32 bits."[1] Essa foi a ferramenta utilizada para a análise do *CIH Vírus*.

- Microsoft Macro Assembler (MASM):

"O Microsoft Macro Assembler (MASM) é um assembler de alto nível x86 para DOS e Microsoft Windows. Versões anteriores eram aplicativos MS-DOS, e o MASM 5.0 possui muitos conjuntos de instruções. O MASM611 e o MASM615 adicionaram a capacidade de produzir programas para aplicativos de console e aplicativos do Windows. O MASM32 é um sistema integrado de arquivos de inclusão, bibliotecas de importação e macros, usado para programação da API do Windows e assembler de 32 bits para a plataforma Windows."[1] Essa foi a ferramenta utilizada para a análise do *rootkit Trojan*.

- IDA version 5.2:

IDA significa “Interactive Disassembler”, é uma ferramenta de análise de código reverso usada para examinar e desmontar programas de computador e encontrar vulnerabilidades de segurança, explorar malware e entender como o código funciona. IDA suporta uma ampla variedade de arquiteturas, como x86, ARM, MIPS e PowerPC, por exemplo. Essa foi uma das ferramentas para efetuar a análise reversa e ver a estrutura visual do *Machine Dog Trojan*.

- OllyDbg version 1.1:

“É um depurador de análise em nível de montador de 32 bits para Windows, capaz de rastrear registros, reconhecer procedimentos, chamadas de API, computadores, tabelas, constantes e strings.”[1]Essa foi a ferramenta utilizada para analisar a string do *Machine Dog Trojan,* para achar informações importantes.

- PE Explorer version 1.98:

“É o programa mais completo para inspecionar o funcionamento interno de vírus, o qual pode examinar arquivos binários PE, realizar análise estática e revelar muitas informações sobre o funcionamento do executável. Após a análise da estrutura PE, os detalhes de importação e exportação do vírus ficam claros.”[1] Essa foi a ferramenta utilizada para obter a tabela de imports do *Machine Dog Trojan*.

- Award Bios Editor version 1.0:

“Ferramenta que permite extrair e substituir as configurações originais da BIOS da placa-mãe pela versão modificada.” [1] Ela foi utilizada para monitorar a BIOS do computador infectado, no processo de análise do *BIOS Trojan*.

1.3 Uso da linguagem Assembly

O conhecimento da linguagem Assembly foi essencial na análise reversa dos códigos dos dois trojans. Principalmente no momento de identificar como era seu funcionamento, já que como eles trabalham em nível de software básico, se escondendo na BIOS do computador, suas ações se baseiam na execução de certas instruções em Assembly, conhecê-las e saber como funcionam é essencial para entender seu funcionamento.

Ainda sim, o papel da linguagem Assembly foi ainda mais importante no momento em desenvolver uma forma de “imunização” e detecção dos web rootkits trojans. Com o conhecimento de como funcionam, foi possível traçar métodos realizando instruções em assembly para encerrar os processos do vírus, tirar a sua autoridade de *host*, deletar seus arquivos e características de dentro do sistema, criar arquivos e pastas de imunização necessários e editar o registro para imunização contra Trojan de internet. Este tipo de estudo só poderia ser conduzido com o Assembly, devido ao teor de software básico de onde o trojan se aloca.

1.4 Resultados e Conclusões

Os resultados do artigo se baseiam na identificação de como os códigos maliciosos funcionam solução proposta apresentada para conter *web rootkit trojans* depois de ser feita uma análise reversa nos códigos maliciosos.

O artigo, conclui que a análise reversa de códigos maliciosos em níveis de software básico como os *web rootkit trojans* é uma alternativa para desenvolver métodos de detecção e de “imunização” com a realização desse experimento na prática.

2. Fichamento do artigo “Opcode and API Based Machine Learning Framework For Malware Classification”

O artigo “*Opcode and API Based Machine Learning Framework For Malware Classification*", foi publicado em 2022 na *2022 2nd International Conference on Intelligent Technologies (CONIT)* , que aconteceu em Hubli, Índia pelos autores Hrishabh Soni, Pushkar Kishore e Durga Prasad Mohapatra.

2.1 Objetivos

O principal objetivo do artigo é demonstrar a efetividade de técnicas de *Machine Learning* para reconhecimento de *Malwares* com o uso de *Datasets* construídos a partir de técnicas de “*n-gram*” em dados de chamadas em APIs do sistema e OPcodes de instruções em assembly realizados pelos códigos maliciosos dentro de um sistema infectado. Assim, contribuindo numa busca em um método efetivo, mas ao mesmo tempo com pouco tempo de treinamento para reconhecimento de *Malwares*.

Desta forma, o FrameWork proposto pelos autores se baseia nos seguintes passos: Desmontar as amostras usando o IDA-pro, extrair as chamadas de API e opcodes de arquivos ASM, Aplicar técnicas de *n-gram* as amostras coletadas de API e opcodes, calcular as frequências de todas as tuplas de *n-gram* e aplicar o TF-IDF, Aplicar o NB, regressão logística, floresta aleatória e classificador de vetores de suporte em conjuntos de dados baseados em API e opcodes e por fim, Selecionar o melhor modelo e calcular as probabilidades de previsão das amostras para ambas as API e opcodes.[2]

2.2 Ferramentas Utilizadas

- Microsoft Malware Classification:

DataSet utilizado para testes, usado para aferir a precisão do método de reconhecimento de *Malwares*.

- IDA-pro:

Disassemble utilizado para transformar os códigos em linguagem de máquina novamente.

- Biblioteca Pyparsing do Python:

Utilizada para construção de código para a aplicação de técnicas de Machine Learning.

2.3 Uso da linguagem Assembly

O uso da linguagem de programação Assembly também é essencial nesse artigo, já que os *Malwares* trabalham em nível de software básico é necessário a extração de suas instruções em *Assembly* para o treinamento e testes dos métodos de *Machine Learning* no processo de identificação dos *Malwares*. Isso só seria possível em Assembly, devido ao baixo nível de operação dos *Malwares* e do objetivo da pesquisa, de identificar os malwares pela execução das instruções .

2.4 Resultados e Conclusões

O artigo conclui mostrando os resultados das técnicas empregadas para detecção de Malwares. Os resultados são muito bons, determinando o sucesso do *Framework* proposto e superando os métodos do estado da arte atual para aquela arquitetura de *FrameWork*, alcançando no geral, a precisão de previsão do quadro final de 99,4%, com uma pontuação F1 de 99%.[2] Os autores deixam claro que no futuro planejam implementar uma arquitetura baseada em *Deep Learning* para classificação de *Malwares* baseado em API e uma nova linha de pesquisa em técnicas de inteligência artificial explicáveis para interpretação dos resultados de de modelos de *machine learning*.

3. Análise comparativa

3.1 Semelhanças

Ambos os artigos tratam de propor soluções e métodos para combater *Malwares*, utilizando de conhecimentos em *Assembly* , baseado nas instruções executadas pelos códigos maliciosos. Além disso,apesar de ambos procuram jeitos diferentes de identificação desses *Malwares*, ambos os estudos só poderiam ser conduzidos com o conhecimento em Assembly, devido ao teor de baixo nível, nível de software básico em que os problemas se estavam. Ambos os artigos também utilizam a ferramenta IDA para realizar os desmontamentos dos códigos em Assembly, com o objetivo de obter as informações sobre o código.

3.2 Diferenças

Embora os objetivos das pesquisas convergem para o mesmo objetivo, de lutar contra os efeitos negativos de um *Malware*, elas fazem isso de maneiras diferentes. No primeiro artigo, apesar de também tratarem de identificação, é muito mais um processo de entender o funcionamento do código malicioso, e bolar um método de imunização. No primeiro artigo, também, o processo de construção da solução envolve muito mais outros aspectos, sem focar muito nas chamadas de API como no segundo, mas sim nas instruções em assembly para identificar os códigos, Import Tables e outros recursos.

Já no segundo, é proposto um *FrameWork* de identificação automática de *Malwares* com técnicas em Inteligência artificial sem nenhuma proposta de imunização, focando bem nas chamadas API do sistema e nas instruções de execução. Algumas ferramentas diferem bastante, apesar de ambos utilizarem o IDA. As diferenças desses artigos se devem devido a um tempo de publicação de 10 anos, que reforça a evolução dos métodos de identificação de Malwares na última década, por fim, ambos procuram objetivos parecidos, mas percorrendo caminhos de conhecimento diferentes, desta forma, seria inevitável as diferenças de abordagem.

References

[1] SONI, Hrishabh; KOSHORE, Pushkar; MOHAPATRA, Durga. Assembly Reverse Analysis on Malicious Code of Web Rootkit Trojan. In: INTERNATIONAL CONFERENCE ON WEB INFORMATION SYSTEMS AND MINING, 5., 2009, Shanghai. Proceedings... Shanghai: IEEE, 2009. p. 361-365. Disponível em: https://ieeexplore.ieee.org/document/5291557. Acesso em: 13 maio 2023.

[2] SHUKLA, Anjali; KUMAR, Ravi. Opcode and API Based Machine Learning Framework For Malware Classification. In: INTERNATIONAL CONFERENCE ON INTELLIGENT TECHNOLOGIES (CONIT), 2., 2022, Karnataka, India. Proceedings... Karnataka: IEEE, 2022. p. 1-6. Disponível em: https://ieeexplore.ieee.org/document/9460969. Acesso em: 13 mai. 2023.