Análise Automatizada de Logs de Instrumentação Binária Dinâmica para Identificação de Comportamentos Evasivos em Executáveis Windows

Margefson M. Barros[1], **Francisco S. S. Neto** [1],
Juan M. DellOso [1], **Yan Soares**[1], **Eduardo L. Feitosa**[1]

[1]Instituto de Computação (IComp) – Universidade Federal do Amazonas (UFAM)
CEP 69.077-000 – Manaus – AM – Brasil
{margefson.barros,francisconeto,delloso.juan,
yan.soares,efeitosa}@icomp.ufam.edu.br

Abstract. *This work proposes FluxTrace, an automated pipeline for the analysis and correlation of multiple Dynamic Binary Instrumentation (DBI) logs associated with the execution of the same Windows Portable Executable (PE) sample. The approach organizes execution traces, extracts behavioral evidence, correlates signals from different instrumentation sources, and maps observed events to evasion-related categories and techniques from the MITRE ATT&CK matrix, producing structured behavioral reports associated with evasive activities. The initial evaluation on 14 potentially evasive PE samples indicated the inference of 11 evasive techniques, demonstrating that the correlation of multiple logs improves the contextual interpretation of evidence observed during dynamic analysis. Comparison with VirusTotal behavioral reports also revealed significant convergence at the parent-technique level despite differences in granularity between the approaches.*

Resumo. *Este trabalho propõe o FluxTrace, um pipeline automatizado para análise e correlação de múltiplos logs de instrumentação binária dinâmica (DBI) associados à execução de uma mesma amostra de arquivos Portable Executable (PE) do Windows. A abordagem organiza rastros de execução, extrai evidências comportamentais, correlaciona sinais provenientes de diferentes fontes de instrumentação e mapeia os eventos observados para categorias relacionadas à evasão e técnicas da matriz MITRE ATT&CK, produzindo relatórios comportamentais estruturados associados a atividades evasivas. A avaliação inicial, conduzida sobre 14 amostras PE potencialmente evasivas, mostrou a presença de 11 técnicas evasão, demonstrando que a correlação entre múltiplos logs melhora a interpretação contextual das evidências observadas durante a análise dinâmica. A comparação com relatórios comportamentais do VirusTotal também revelou convergência significativa em nível de técnica-pai, apesar das diferenças de granularidade entre as abordagens.*

1. Introdução

A análise dinâmica de *malware* constitui uma etapa fundamental para compreender o comportamento de programas potencialmente maliciosos durante a execução. Diferentemente da análise estática, que examina o binário sem executá-lo, a análise dinâmica

permite observar ações realizadas em tempo de execução, como chamadas de API, acesso a arquivos, consultas ao registro, criação de processos, carregamento de bibliotecas e interações com o sistema operacional [Egele et al. 2012, Greamo 2011]. No contexto de arquivos Windows no formato *Portable Executable* (PE), essa observação torna-se particularmente relevante, pois muitas amostras ocultam funcionalidades por meio de empacotamento, ofuscação, resolução dinâmica de funções e execução condicionada ao ambiente [You and Yim 2010, Ugarte-Pedrero et al. 2019].

Nesse contexto, uma estratégia utilizada para analisar a execução de programas é a *Dynamic Binary Instrumentation* (DBI). A DBI permite inserir rotinas de monitoramento durante a execução do binário sem exigir modificações prévias no executável [Nethercote and Seward 2007, Luk et al. 2005]. Ferramentas construídas sobre *frameworks* DBI, como Intel Pin, DynamoRIO e Valgrind, possibilitam a coleta de informações de baixo nível e de eventos comportamentais associados à execução do programa [Luk et al. 2005, Bruening et al. 2004, Nethercote and Seward 2007]. Esses eventos podem ser organizados em diferentes arquivos de log, nos quais cada registro tende a representar uma perspectiva específica da execução monitorada, incluindo chamadas de API, instruções executadas, módulos carregados, fluxos de controle, consultas ao ambiente, acessos ao sistema de arquivos e interações entre processos.

Entretanto, à medida que ambientes instrumentados passaram a ser amplamente utilizados na análise dinâmica, aplicações maliciosas modernas passaram a incorporar mecanismos capazes de identificar *sandboxes*, máquinas virtuais, depuradores e ambientes instrumentados [Kirat et al. 2011, Balzarotti et al. 2010, Polino et al. 2017]. Quando sinais associados à análise são detectados, a amostra pode reduzir sua atividade, atrasar a execução de rotinas maliciosas, modificar o fluxo de controle, encerrar o processo prematuramente ou simular comportamento benigno [Sun et al. 2016, Zhechev 2018]. Como consequência, mecanismos evasivos podem comprometer a qualidade dos rastros produzidos durante a análise dinâmica, gerando evidências incompletas, ambíguas ou aparentemente benignas.

Além disso, a análise isolada de um único log pode ser insuficiente para caracterizar evasão. Uma chamada de API específica pode representar comportamento legítimo em determinados contextos; entretanto, quando combinada com consultas a artefatos de virtualização, medições temporais, enumeração de processos, busca por módulos de depuração e encerramento prematuro da execução, o conjunto de evidências passa a sugerir comportamento evasivo [Rodriguez et al. 2016, Polino et al. 2017]. Nesse cenário, diferentes logs produzidos durante a execução de um mesmo executável representam fontes complementares de evidência, cuja correlação pode fornecer uma visão mais abrangente e consistente do comportamento observado. Apesar disso, grande parte das abordagens existentes concentra-se na coleta ou inspeção isolada dos rastros produzidos pela instrumentação, dificultando a interpretação integrada dos eventos observados durante a execução monitorada.

Diante desse contexto, este trabalho propõe o FluxTrace[1], um pipeline automatizado voltado à extração, organização e interpretação de evidências comportamentais presentes em logs produzidos por análises baseadas em DBI. A proposta parte da premissa

[1]O pipeline FluxTrace está disponível publicamente em: **`https://github.com/fluxtrace`**

de que diferentes arquivos de log representam visões complementares da execução e que a integração dessas perspectivas pode auxiliar na identificação de sinais associados à evasão. Para isso, o pipeline recebe diretórios de logs associados a uma amostra, normaliza os dados coletados, aplica regras de detecção, classifica evidências em categorias comportamentais e gera relatórios analíticos voltados ao apoio de pesquisadores e analistas de segurança.

De forma resumida, esta pesquisa contribui com: (i) a correlação automatizada de múltiplos logs produzidos por DBI; (ii) a organização de evidências em categorias semânticas associadas à evasão; (iii) o mapeamento das evidências observadas para técnicas da matriz MITRE ATT&CK; (iv) a comparação das inferências produzidas com relatórios comportamentais públicos do VirusTotal; e (v) a geração de relatórios estruturados voltados à interpretação de comportamentos potencialmente evasivos.

A principal contribuição deste trabalho consiste em transformar logs brutos de instrumentação em evidências estruturadas de comportamento evasivo. Para isso, a abordagem organiza os sinais observados em categorias interpretáveis, registra o encadeamento lógico associado às inferências produzidas e permite que os resultados sejam posteriormente utilizados tanto em análises manuais quanto em experimentos quantitativos. Por fim, o artigo apresenta a arquitetura inicial do pipeline proposto e estabelece as bases metodológicas para sua futura avaliação experimental.

2. Fundamentação Teórica

2.1. Comportamentos Evasivos

Comportamentos evasivos correspondem a estratégias empregadas por *malwares* para dificultar sua análise, classificação ou detecção em ambientes monitorados [Botacin et al. 2017, Polino et al. 2017]. Entre os mecanismos mais comuns estão a verificação da presença de depuradores, a busca por artefatos associados a máquinas virtuais, a identificação de processos relacionados a *sandboxes*, o uso de chamadas relacionadas à temporização e a inspeção de características do ambiente de execução, como nomes de usuário, nomes de máquina, quantidade de núcleos de CPU e módulos carregados [Shields 2009, Issa 2012, Shi and Mirkovic 2017]. Em muitos casos, aplicações maliciosas também podem interromper ou modificar seu fluxo de execução quando o ambiente aparenta ser artificial ou monitorado.

Essas estratégias permitem que *malwares* adaptem seu comportamento de acordo com o contexto observado, dificultando a coleta consistente de evidências durante a análise dinâmica [Or-Meir et al. 2019, Apostolopoulos et al. 2021]. Como consequência, sinais associados à evasão frequentemente aparecem distribuídos entre diferentes eventos registrados ao longo da execução, tornando limitada a interpretação isolada de chamadas de API, instruções ou acessos específicos observados durante o monitoramento.

Neste trabalho, a evasão é tratada como um fenômeno observável por meio de evidências comportamentais extraídas dos logs de instrumentação. Embora uma evidência isolada possa não ser conclusiva, a combinação de múltiplos sinais pode aumentar a confiança das inferências produzidas. Dessa forma, o pipeline proposto organiza os eventos monitorados em categorias semânticas e registra quais evidências sustentam cada inferência comportamental, permitindo estruturar os rastros observados durante a execução monitorada.

2.2. Instrumentação Binária Dinâmica

A *Dynamic Binary Instrumentation* (DBI) consiste em uma técnica que permite inserir instruções adicionais no código de um programa durante sua execução, sem a necessidade de acesso ao código-fonte ou recompilação do executável original. Essa abordagem vem sendo aplicada em diferentes contextos, incluindo engenharia reversa, análise de *malwares*, monitoramento de desempenho e avaliação de segurança [Wu et al. 2023, Nethercote and Seward 2007, Zhechev 2018].

Diferentemente das estratégias baseadas exclusivamente em análise estática, a DBI possibilita modificar dinamicamente o fluxo de execução do programa, permitindo a coleta em tempo real de informações associadas ao estado interno do processo monitorado. Entre suas capacidades estão a leitura e escrita em registradores, a inspeção de regiões de memória e a interceptação de chamadas ao sistema operacional, funcionalidades frequentemente utilizadas na análise de executáveis potencialmente maliciosos.

Nesse contexto, Ferramentas baseadas em DBI passaram a ser empregadas no monitoramento de comportamentos observados durante a execução de aplicações instrumentadas. Entretanto, apesar das vantagens oferecidas por essa técnica, *frameworks* DBI apresentam limitações relacionadas à análise de programas evasivos. Em muitos casos, aplicações maliciosas conseguem identificar ambientes instrumentados por meio de verificações realizadas durante a execução, comprometendo a observação de comportamentos potencialmente relevantes [Sun et al. 2016, Zhechev 2018]. Como consequência, estratégias evasivas podem reduzir a eficácia da análise dinâmica e dificultar a obtenção de rastros consistentes em ambientes baseados em DBI [Polino et al. 2017, Rodriguez et al. 2016, Neto et al. 2025].

2.3. Instrumentadores DBI

Os primeiros instrumentadores DBI surgiram no início dos anos 2000 a partir de iniciativas voltadas ao desenvolvimento de ferramentas extensíveis capazes de abstrair a complexidade da instrumentação e reduzir o impacto de desempenho durante a análise dinâmica. Desde então, diferentes *frameworks* DBI passaram a ser utilizados em atividades relacionadas à engenharia reversa, análise de *malwares*, depuração e monitoramento de segurança.

Instrumentadores como Valgrind [Nethercote and Seward 2007], DynamoRIO [Bruening et al. 2004], Intel Pin [Luk et al. 2005] e Frida [Frida Developers 2014] permitem o desenvolvimento de aplicações capazes de definir pontos de instrumentação e ações executadas durante a análise do binário-alvo. De maneira geral, esses *frameworks* possibilitam monitorar o comportamento do programa em tempo de execução, interceptando chamadas de funções, acompanhando o fluxo de controle, analisando acessos à memória e coletando informações sobre o estado interno da aplicação instrumentada.

2.4. Logs de Instrumentação Dinâmica

Ferramentas baseadas em DBI frequentemente produzem múltiplos logs associados ao monitoramento da aplicação instrumentada, incluindo rastros de chamadas de API, instruções executadas, acessos à memória e informações relacionadas ao fluxo de controle. Esses registros representam diferentes perspectivas do comportamento observado durante

a execução do programa e permitem analisar eventos produzidos em múltiplas camadas da aplicação monitorada.

De maneira geral, logs de chamadas de funções permitem observar interações com o sistema operacional e bibliotecas externas, enquanto rastros de instruções e acessos à memória possibilitam identificar operações de baixo nível relacionadas ao comportamento interno da aplicação. Como consequência, diferentes técnicas potencialmente evasivas podem manifestar sinais distintos em cada fonte de evidência produzida pela instrumentação dinâmica.

Neste trabalho, utilizam-se os logs produzidos pela ferramenta Contradef [Campelo et al. 2025], como fontes complementares de evidência para identificação de comportamentos potencialmente evasivos. A proposta parte da premissa de que a correlação entre múltiplos logs pode fornecer uma representação mais consistente do comportamento monitorado do que a análise isolada de eventos individuais. Nesse contexto, eventos aparentemente benignos podem tornar-se relevantes quando analisados em conjunto com outras evidências observadas ao longo da execução instrumentada.

Além disso, devido ao elevado volume de informações produzido por ambientes DBI, o pipeline proposto realiza etapas de filtragem, organização e correlação das evidências coletadas, buscando reduzir ambiguidades e estruturar os eventos monitorados em categorias comportamentais mais interpretáveis para o analista. Dessa forma, a abordagem procura transformar logs heterogêneos e de baixo nível em evidências estruturadas capazes de auxiliar a interpretação de comportamentos associados à evasão.

3. Trabalhos Relacionados

A análise dinâmica de executáveis Windows constitui uma abordagem importante para compreender o comportamento de aplicações potencialmente maliciosas durante a execução. Diferentemente da análise estática, abordagens dinâmicas permitem observar interações com o sistema operacional, incluindo chamadas de API, manipulação de memória, criação de processos, alterações no registro e operações em arquivos [Egele et al. 2012, Or-Meir et al. 2019]. Nesse contexto, mecanismos de monitoramento e instrumentação tornaram-se componentes relevantes para a coleta de evidências comportamentais associadas à execução de *malwares* modernos.

Entre os primeiros trabalhos voltados à análise comportamental baseada em eventos, [Seifert et al. 2007] apresentam o Capture, uma ferramenta destinada ao registro de alterações realizadas no sistema durante a execução de aplicações em ambientes Windows. O trabalho evidencia desafios relacionados ao elevado volume de eventos produzidos durante a análise dinâmica, dificultando distinguir ações legítimas de comportamentos potencialmente maliciosos.

Na sequência, [Shields 2009] descrevem diferentes mecanismos de anti-debugging utilizados para dificultar engenharia reversa, incluindo verificações baseadas em APIs do Windows, temporização e inspeção de registradores. Posteriormente, [Issa 2012] investigam mecanismos de anti-emulação e anti-máquina virtual empregados por famílias modernas de *malware*, demonstrando como diferenças no estado inicial do ambiente podem ser exploradas para detectar sistemas virtualizados.

No contexto de levantamentos sobre análise dinâmica automatizada,

[Egele et al. 2012] descrevem mecanismos utilizados para executar amostras suspeitas em ambientes monitorados e registrar eventos produzidos durante a execução. Os autores destacam que a análise dinâmica permite observar ações frequentemente ocultadas por empacotamento, ofuscação e carregamento dinâmico de código.

Com o avanço das técnicas evasivas, estudos passaram a investigar mecanismos específicos de anti-análise. [Botacin et al. 2017] discutem técnicas de anti-análise e anti-anti-análise utilizadas para detectar ambientes virtualizados, depuradores e ferramentas de monitoramento. No mesmo período, [Polino et al. 2017] demonstram que aplicações evasivas podem identificar artefatos associados ao Intel Pin e modificar seu fluxo de execução para evitar a exposição de comportamentos potencialmente maliciosos. Além disso, os autores apresentam o Arancino, uma plataforma baseada em DBI voltada à identificação de técnicas de anti-instrumentação.

Também em 2017, [Shi and Mirkovic 2017] apresentam o Apate, uma abordagem voltada à ocultação de depuradores frente a técnicas modernas de anti-debugging. O trabalho demonstra que aplicações evasivas frequentemente empregam múltiplas estratégias simultaneamente, dificultando a observação consistente do comportamento monitorado.

Mais recentemente, [Or-Meir et al. 2019] discutem mecanismos modernos de monitoramento comportamental, análise em *sandbox* e observação multicamada, ressaltando que diferentes ferramentas tendem a capturar perspectivas distintas da execução, tornando necessária a combinação de múltiplas fontes de evidência. De forma semelhante, [Apostolopoulos et al. 2021] discutem mecanismos voltados à remoção de *hooks* empregados por ferramentas modernas de análise dinâmica, evidenciando limitações presentes em soluções tradicionais de monitoramento.

Embora os trabalhos apresentados avancem significativamente na instrumentação da execução, identificação de evasão e monitoramento comportamental, grande parte das abordagens concentra-se na coleta de rastros ou na detecção de técnicas evasivas específicas. Em contrapartida, este trabalho direciona seu foco para a análise automatizada e correlação desses múltiplos logs durante a execução de um mesmo executável Windows, buscando transformar eventos heterogêneos em evidências estruturadas de comportamento evasivo.

4. Pipeline Proposto

A Figura 1 apresenta uma visão geral do pipeline proposta (FluxTrace), incluindo as etapas de processamento dos logs, extração de evidências, correlação entre múltiplas fontes de rastreamento e geração dos relatórios analíticos.

4.1. Processamento e Redução dos Logs

Os logs produzidos durante a instrumentação DBI são organizados por amostra e processados de forma incremental. O pipeline identifica automaticamente registros provenientes do **Rastreamento de Chamadas Função** (*TraceFcnCall*), **Rasteramento de Instruções** (*TraceInstructions*), **Rastreamento de Memória** (*TraceMemory*), **Rastreamento de Desmontagem** (*TraceDisassembly*) e **Intercecptador de Funções** (*FunctionInterceptor*).

Uma vez que os logs produzidos por ambientes DBI podem atingir milhões de linhas, o pipeline aplica um mecanismo determinístico de redução heurística voltado à

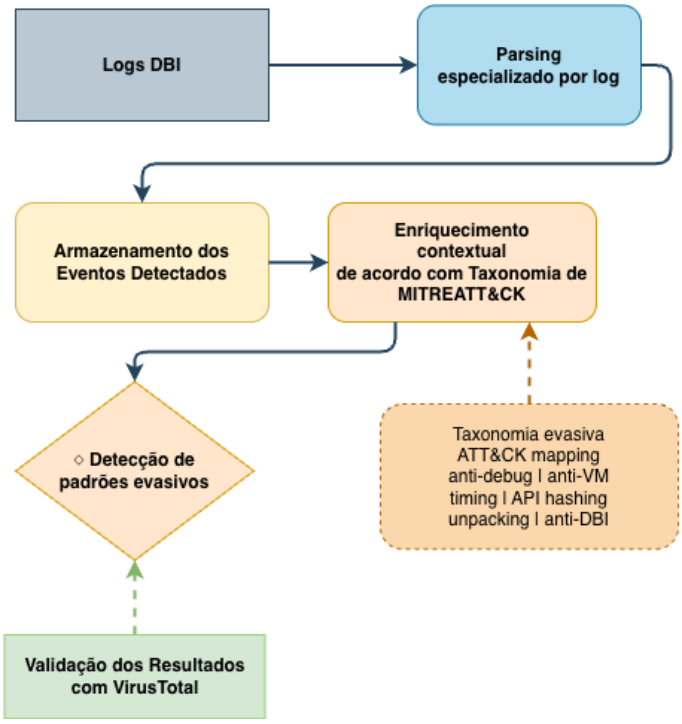

Figura 1. Visão geral do pipeline FluxTrace para análise e correlação de logs DBI.

preservação de eventos potencialmente relevantes para análise comportamental. A abordagem mantém janelas de contexto ao redor de eventos suspeitos, reduzindo significativamente o volume bruto de dados processados sem descartar evidências potencialmente associadas à evasão.

A seleção heurística considera eventos relacionados a técnicas frequentemente associadas à anti-análise, incluindo verificações de depuração e virtualização, resolução dinâmica de APIs, manipulação de memória executável, temporização, injeção de código e acessos ao PEB/TEB[2].

Para avaliar o impacto da redução aplicada aos logs, utiliza-se como métrica principal a taxa de redução por número de linhas:

$$\rho_{lin} = 100 \cdot \frac{N - |K|}{N}$$

onde N representa o número total de linhas do log original e $|K|$ corresponde ao conjunto de linhas preservadas após a etapa de filtragem.

4.2. Extração e Classificação de Evidências

Após a filtragem, os eventos preservados são organizados em categorias semânticas associadas à evasão, incluindo *Anti-debugging*, *Anti-VM*, *Timing Evasion*, *Dynamic API Resolution*, *Code Injection*, *Environment Inspection* e *Process Manipulation*.

[2]PEB (*Process Environment Block*) e TEB (*Thread Environment Block*) são estruturas internas do Windows utilizadas para armazenar informações sobre processos e *threads* em execução.

As evidências extraídas são associadas à matriz MITRE ATT&CK, particularmente à tática TA0005 (*Defense Evasion*), permitindo estruturar os comportamentos observados durante a análise dinâmica. Opcionalmente, os resultados também podem ser enriquecidos com informações públicas provenientes do VirusTotal utilizando o SHA-256 da amostra analisada.

4.3. Correlação entre Logs

Os diferentes logs produzidos pela instrumentação representam perspectivas complementares do comportamento monitorado. Nesse contexto, o pipeline realiza a correlação entre múltiplas fontes de evidência com o objetivo de produzir inferências mais consistentes sobre comportamentos potencialmente evasivos. De maneira geral, a correlação considera fatores como recorrência de eventos, diversidade dos módulos de rastreamento envolvidos, proximidade temporal entre evidências e categorias comportamentais frequentemente associadas a técnicas de evasão descritas na literatura.

Em vez de interpretar eventos isoladamente, o pipeline busca identificar padrões compostos de execução a partir da combinação entre diferentes camadas de observabilidade produzidas pela instrumentação DBI. Verificações de depuração, por exemplo, podem ser identificadas tanto por chamadas diretas de APIs registradas no log de *TraceFcnCall* quanto por acessos ao PEB observados nos logs *TraceInstructions* e *TraceMemory*. De forma semelhante, técnicas de *timing evasion* podem ser inferidas pela correlação entre chamadas de temporização, leituras recorrentes da estrutura *KUSER_SHARED_DATA* e padrões repetitivos de consulta temporal.

Além disso, a utilização conjunta dos diferentes logs permite reduzir ambiguidades inerentes à interpretação de artefatos isolados. Funções como *LoadLibrary*, *VirtualAlloc* e *GetTickCount*, por exemplo, podem estar presentes tanto em aplicações legítimas quanto em programas maliciosos. Dessa forma, a análise considera não apenas a ocorrência individual dessas evidências, mas também sua frequência, contexto de execução e associação com outros eventos observados simultaneamente.

Como resultado, o pipeline favorece uma reconstrução mais contextualizada do comportamento instrumentado, permitindo identificar combinações de eventos potencialmente associadas a técnicas de anti-instrumentação, evasão de análise e mecanismos de proteção em tempo de execução.

4.4. Relatórios Gerados

Ao final do processamento, o pipeline produz relatórios estruturados contendo categorias comportamentais identificadas, evidências correlacionadas, rastreabilidade para os logs originais e indicadores associados a possíveis técnicas de evasão observadas durante a execução instrumentada. O objetivo desses relatórios é reduzir o elevado volume de dados normalmente produzido pela instrumentação DBI, organizando os eventos coletados em representações mais interpretáveis para o analista.

Os relatórios consolidam informações dos logs, integrando evidências relacionadas a chamadas de APIs, acessos à memória, rastreamento de instruções, resolução dinâmica de funções e verificações de ambiente ou temporização. De maneira geral, cada registro mantém informações como módulo de origem, trechos relevantes dos logs, funções

observadas, endereços monitorados e sequências de execução associadas às inferências produzidas pelo pipeline.

Essa estratégia permite preservar a rastreabilidade das análises realizadas, possibilitando auditoria manual, validação dos resultados e refinamento posterior das regras de correlação empregadas pelo sistema. Além disso, a consolidação dos eventos em categorias comportamentais reduz ambiguidades decorrentes da interpretação isolada de APIs ou instruções específicas, favorecendo uma análise mais contextualizada do comportamento monitorado.

Como resultado, os relatórios produzidos pelo pipeline auxiliam especialistas na investigação de executáveis potencialmente evasivos, permitindo identificar padrões recorrentes associados a técnicas de anti-instrumentação, evasão temporal, inspeção de ambiente e mecanismos de proteção em tempo de execução.

5. Planejamento Experimental

A avaliação da abordagem proposta foi conduzida sobre um conjunto de 14 amostras PE de Windows previamente identificadas como potencialmente evasivas, obtidas a partir da plataforma MalwareBazaar[3] e acompanhadas de seus respectivos conjuntos de logs de instrumentação produzidos pela ferramenta Contradef. Todas as execuções geraram logs de *TraceFcnCall*, *TraceInstructions*, *TraceMemory*, *TraceDisassembly* e *FunctionInterceptor*.

O objetivo da avaliação foi verificar: (i) a capacidade do pipeline de extrair, categorizar e mapear evidências comportamentais para a tática TA0005 (*Defense Evasion*) da matriz MITRE ATT&CK; e (ii) a aderência das técnicas inferidas em relação a uma referência externa pública representada pelos relatórios comportamentais do VirusTotal (VT).

5.1. Dados de Entrada

Os dados de entrada consistem em diretórios de logs, nos quais cada diretório representa uma amostra PE e contém os arquivos produzidos pela instrumentação dinâmica. A metodologia adotada considera conjuntos equivalentes de logs para todas as amostras analisadas, registrando eventuais ausências de fontes para evitar interpretações indevidas durante o processamento.

Cada amostra foi identificada por seu *hash* SHA-256 e processada de forma independente pelo pipeline. Para cada execução, o sistema produziu um conjunto de evidências organizadas em categorias semânticas, incluindo *anti-debugging*, *anti-VM*, *timing evasion*, resolução dinâmica de APIs, injeção de código, inspeção de ambiente e manipulação de processos. A partir dessas categorias, foi derivado o subconjunto de técnicas MITRE ATT&CK pertencentes à tática TA0005.

Em paralelo, foram coletados os relatórios comportamentais públicos do VirusTotal associados aos mesmos SHA-256, dos quais foram extraídos: (i) o conjunto completo de técnicas ATT&CK reportadas (*virustotal_behaviour_mitre*); e (ii) o subconjunto filtrado pertencente exclusivamente à tática TA0005 (*TA0005 VirusTotal*), utilizado como base de comparação direta.

[3] `https://bazaar.abuse.ch/`

5.2. Critérios de Avaliação

Para cada amostra, foram calculadas três métricas sobre o conjunto união de técnicas TA0005 reportadas pelas duas fontes: **Match (%)**, correspondente à proporção de técnicas detectadas simultaneamente por FluxTrace e VirusTotal; **Apenas FluxTrace (%)**, relacionada às técnicas reportadas exclusivamente pelo pipeline proposto; e **Apenas VirusTotal (%)**, referente às técnicas reportadas exclusivamente pelo VirusTotal.

Adicionalmente, foram contabilizadas a frequência absoluta e relativa de cada técnica inferida por ambas as fontes, bem como o conjunto total de técnicas únicas detectadas, permitindo caracterizar o perfil de cobertura de cada abordagem.

6. Resultados e Discussão

Esta seção apresenta os resultados obtidos sobre o conjunto de 14 amostras avaliadas, considerando: (i) a cobertura das técnicas inferidas e o perfil de detecção do pipeline; (ii) a análise comparativa com o VirusTotal, incluindo correspondência exata e convergência em nível de técnica-pai; e (iii) o impacto da correlação entre múltiplos logs DBI sobre a qualidade das evidências geradas.

6.1. Cobertura e Perfil de Detecção

O pipeline FluxTrace inferiu **11 técnicas únicas** pertencentes à tática TA0005 (*Defense Evasion*) ao longo do conjunto avaliado. A Tabela 1 apresenta a frequência de ocorrência de cada técnica.

Tabela 1. Frequência das técnicas TA0005 inferidas pelo FluxTrace ($n = 14$).

ID	Técnica	Frequência
T1027	Obfuscated Files or Information	14/14 (100,0%)
T1027.007	Dynamic API Resolution	14/14 (100,0%)
T1055	Process Injection	14/14 (100,0%)
T1070	Indicator Removal	14/14 (100,0%)
T1070.004	File Deletion	14/14 (100,0%)
T1497.003	Time Based Checks	14/14 (100,0%)
T1622	Debugger Evasion	14/14 (100,0%)
T1678	Delay Execution	14/14 (100,0%)
T1055.002	Portable Executable Injection	9/14 (64,3%)
T1497.001	System Checks	7/14 (50,0%)
T1055.001	Dynamic-link Library Injection	5/14 (35,7%)

Oito (8) técnicas foram observadas em 100% das amostras analisadas, enquanto as três restantes correspondem a variantes específicas de *Process Injection* (T1055.001 e T1055.002) e a uma subtécnica adicional de *Virtualization/Sandbox Evasion* (T1497.001), cuja ocorrência depende do conjunto particular de chamadas observadas durante a execução.

A presença consistente dessas técnicas é coerente com a forma como o pipeline foi construído. As regras heurísticas aplicadas sobre os logs *TraceFcnCall*, *TraceInstructions* e *TraceMemory* apresentam alta sensibilidade para sinais associados à resolução dinâmica de APIs, manipulação de memória executável, consultas temporais e verificações típicas de *debugger checks*. Como consequência, o pipeline opera principalmente como um detector de baixo nível, sensível a comportamentos elementares observáveis durante a instrumentação, e não como um classificador baseado em assinaturas de famílias específicas de *malware*.

6.2. Comparação com o VirusTotal

A Tabela 2 resume as métricas calculadas sobre o conjunto avaliado.

Tabela 2. Resumo da comparação entre FluxTrace e VirusTotal ($n = 14$).

Métrica	Média	Mediana	Mínimo	Máximo	Desvio-padrão
Match (%)	9,39	9,10	0,00	20,00	5,13
Apenas FluxTrace (%)	63,21	66,70	30,00	100,00	21,64
Apenas VirusTotal (%)	27,39	26,80	0,00	54,50	18,23

A interseção média (*Match*) entre as técnicas TA0005 reportadas por FluxTrace e VirusTotal foi de **9,39%** (mediana 9,10%; desvio-padrão 5,13%). A proporção média de técnicas reportadas exclusivamente por FluxTrace foi de **63,21%**, enquanto o VirusTotal apresentou média de **27,39%** de técnicas exclusivas. Em duas amostras (`49aa7438...` e `1db3df87...`), a interseção foi nula; na segunda, o VirusTotal não reportou qualquer técnica TA0005. A baixa interseção observada deve ser interpretada com cautela e não corresponde, isoladamente, a uma medida de acurácia do pipeline. Três fatores principais ajudam a explicar esse resultado:

Primeiro, granularidade distinta. As técnicas TA0005 reportadas pelo VirusTotal concentram-se em rótulos de nível mais alto, predominantemente *T1027* (*Obfuscated Files or Information*, presente em 85,7% das amostras), *T1027.002* (*Software Packing*, 71,4%) e *T1497* (*Virtualization/Sandbox Evasion*, 57,1%). Em contraste, o FluxTrace inferiu subtécnicas mais específicas observáveis diretamente nos logs de instrumentação, como *T1027.007* (*Dynamic API Resolution*), *T1497.003* (*Time Based Checks*), *T1678* (*Delay Execution*) e *T1055.002* (*Portable Executable Injection*). Assim, ambas as fontes frequentemente descrevem o mesmo fenômeno em níveis distintos de abstração.

Para evidenciar esse efeito, a Tabela 3 apresenta a comparação por técnica-pai, agregando subtécnicas ao seu respectivo identificador genérico. Em nível agregado, a convergência aumenta significativamente para *T1027* (100,0% vs 85,7%) e *T1497* (100,0% vs 57,1%), embora permaneçam divergências relevantes em *T1055* e *T1070*, técnicas raramente reportadas diretamente pelo VirusTotal no recorte TA0005.

Tabela 3. Convergência por técnica-pai (proporção de amostras em que ao menos uma subtécnica do grupo foi reportada).

Técnica-pai	Descrição	FluxTrace	VirusTotal
T1027	Obfuscated Files or Information	100,0%	85,7%
T1055	Process Injection	100,0%	35,7%
T1070	Indicator Removal	100,0%	14,3%
T1497	Virtualization/Sandbox Evasion	100,0%	57,1%

Segundo, fontes complementares. O VirusTotal agrega resultados provenientes de múltiplos *sandboxes* e mecanismos de análise, capturando uma visão mais ampla do ciclo de vida do *malware*, incluindo persistência, exfiltração e comportamentos pós-execução que estão fora do escopo da tática TA0005. Além disso, as técnicas reportadas pelos *sandboxes* frequentemente derivam de heurísticas próprias e padrões comportamentais agregados, distintos dos sinais de baixo nível observados pela instrumentação DBI.

Terceiro, especialização do pipeline. O FluxTrace foi projetado para identificar evidências evasivas manifestadas no nível das instruções e chamadas de API. Técnicas

como *Masquerading* (T1036), *Hijack Execution Flow* (T1574.002) e *Impair Defenses* (T1562), observadas no VirusTotal, exigem visibilidade sobre artefatos persistentes do sistema de arquivos, do registro ou sobre o uso de binários legítimos do sistema operacional, escopos não contemplados pela instrumentação atual.

Dois casos extremos ilustram a complementaridade da abordagem. Na amostra `1db3df87...`, o VirusTotal não reportou qualquer técnica da tática TA0005 (*Defense Evasion*), enquanto o FluxTrace identificou nove técnicas a partir dos logs de instrumentação, evidenciando que as assinaturas comportamentais dos *sandboxes* consultados não rotularam tais técnicas para a execução analisada. Em contraste, a amostra `a0aeb837...` apresentou 54,5% de técnicas exclusivas no VirusTotal, evidenciando comportamentos como *Masquerading*, *Hidden Window*, *DLL Side-Loading* e *System Binary Proxy Execution*, categorias que o pipeline ainda não está aparelhado para inferir.

6.3. Impacto da Correlação entre Logs

A correlação entre os cinco logs de instrumentação utilizados mostrou-se determinante para fundamentar as técnicas inferidas. Em particular, a técnica T1027.007 (*Dynamic API Resolution*) foi inferida apenas quando evidências oriundas do log *TraceFcnCall* (chamadas a `GetProcAddress` e `LoadLibrary`) co-ocorreram com evidências do log *TraceInstructions*, indicando acessos ao PEB/TEB e chamadas indiretas em regiões sem símbolo observadas pelo *TraceDisassembly*. De forma semelhante, a inferência da técnica T1055.002 (*Portable Executable Injection*) dependeu da combinação entre evidências do log *TraceMemory* (alocações com `PAGE_EXECUTE_READWRITE`) e do log *TraceFcnCall* (chamadas a `VirtualAlloc`, `WriteProcessMemory` e `CreateRemoteThread`).

Em contraste, técnicas como T1622 (*Debugger Evasion*) puderam ser inferidas inicialmente a partir de uma única fonte, especialmente por chamadas a `IsDebuggerPresent` e `NtQueryInformationProcess` registradas pelo log *TraceFcnCall*. Entretanto, a inspeção cruzada com evidências provenientes do log *TraceInstructions* aumentou a robustez da inferência, reduzindo falsos positivos em situações onde essas chamadas ocorriam em rotinas legítimas de inicialização de bibliotecas. Esse resultado é consistente com a hipótese central deste trabalho: a análise isolada de um único log tende a ser limitada, enquanto a correlação entre múltiplas fontes amplia o poder explicativo das evidências extraídas.

6.4. Síntese

Os resultados sugerem que o pipeline proposto contribui com uma camada de análise distinta e complementar àquelas oferecidas por serviços públicos de análise comportamental. Embora a interseção média em correspondência exata tenha sido de apenas 9,39%, a convergência observada em nível de técnica-pai (Tabela 3) indica acordo significativo quando subtécnicas relacionadas são agregadas em categorias mais amplas.

Por outro lado, os resultados também evidenciam que o conjunto atual de regras é fortemente influenciado por um núcleo de oito técnicas detectadas em 100% das amostras avaliadas. Esse comportamento indica a necessidade de investigações adicionais sobre a sensibilidade e a especificidade individual das heurísticas empregadas antes da extrapolação dos resultados para conjuntos maiores ou para amostras não evasivas.

Tabela 4. Resumo dos principais resultados observados.

Aspecto	Resultado Principal
Técnicas TA0005 inferidas	11
Técnicas recorrentes	8
Match médio com VT	9,39%
Convergência por técnica-pai	Elevada
Logs mais relevantes	*TraceFcnCall* + *TraceInstructions*
Principal limitação	Dependência de heurísticas

7. Limitações e Ameaças à Validade

A proposta depende diretamente da qualidade e da abrangência dos logs produzidos pela ferramenta de DBI utilizada durante a instrumentação. Caso determinados eventos não sejam monitorados ou registrados, o pipeline não poderá inferir evidências relacionadas a esses comportamentos. Além disso, a ausência de uma evidência no log não implica necessariamente ausência do comportamento na amostra analisada.

Outra limitação está associada à natureza interpretável e baseada em regras da abordagem proposta. Embora regras heurísticas favoreçam explicabilidade, rastreabilidade e auditoria das inferências produzidas, elas podem não capturar variantes evasivas ainda não previstas pelo conjunto atual de categorias e padrões comportamentais. Dessa forma, o conjunto de regras empregado deve ser tratado como incremental e sujeito a refinamento contínuo à medida que novas amostras forem analisadas.

Também existe ameaça à validade externa dos resultados. As evidências observadas dependem da coleção de amostras utilizada, da configuração do ambiente de instrumentação e dos mecanismos específicos de monitoramento disponibilizados pela ferramenta DBI empregada. Assim, resultados obtidos em um determinado cenário experimental podem não generalizar diretamente para outros ambientes, *sandboxes*, coleções de amostras ou famílias de *malware*. Para reduzir esse risco, a metodologia adotada documenta o ambiente de execução, os tipos de logs disponíveis e os critérios empregados na categorização das evidências inferidas.

8. Conclusão

Este trabalho apresentou um pipeline automatizado para análise e correlação de múltiplos logs de instrumentação binária dinâmica, voltado à identificação de comportamentos evasivos em executáveis Windows. A proposta organiza rastros heterogêneos provenientes de logs de **Rastreamento de Chamadas Função** (*TraceFcnCall*), **Rasteramento de Instruções** (*TraceInstructions*), **Rastreamento de Memória** (*TraceMemory*), **Rastreamento de Desmontagem** (*TraceDisassembly*) e **Intercecptador de Funções** (*FunctionInterceptor*) em evidências estruturadas, categorizando-as em classes semânticas associadas à tática *Defense Evasion* (TA0005) da matriz MITRE ATT&CK e gerando relatórios analíticos rastreáveis.

A avaliação inicial, conduzida sobre 14 amostras PE potencialmente evasivas, indicou que o pipeline é capaz de inferir consistentemente um conjunto de 11 técnicas TA0005 a partir dos logs de execução, sendo oito delas observadas em 100% das amostras analisadas. A comparação com os relatórios comportamentais do VirusTotal revelou uma interseção média de 9,39% em correspondência exata de identificadores, percen-

tual que aumenta substancialmente quando a análise é realizada em nível de técnica-pai, conforme discutido na Seção 6. Esse comportamento é explicado pela diferença de granularidade entre as duas abordagens, uma vez que o FluxTrace opera sobre subtécnicas diretamente observáveis na instrumentação DBI, enquanto o VirusTotal tende a produzir rótulos comportamentais mais agregados derivados de *sandboxes*.

Os resultados também demonstraram que a correlação entre múltiplas fontes de log é fundamental para sustentar inferências relacionadas a resolução dinâmica de APIs, injeção em processos e verificações de ambiente, ampliando o contexto das evidências observadas durante a execução instrumentada. Ao mesmo tempo, o estudo evidencia limitações importantes da abordagem atual. O conjunto experimental é reduzido e composto exclusivamente por amostras potencialmente evasivas, o que limita avaliações mais robustas de especificidade; a ausência de uma base anotada por especialistas restringe o cálculo de métricas tradicionais de classificação; e a predominância de um núcleo de oito técnicas detectadas em todas as amostras indica que parte das heurísticas empregadas apresenta elevada sensibilidade e ainda necessita calibração.

Como trabalhos futuros, pretende-se: (i) expandir o conjunto experimental com amostras benignas e famílias distintas de programas maliciosos,; (ii) construir uma base anotada por especialistas para cálculo de métricas como precisão, revocação e medida F1; (iii) refinar as heurísticas utilizadas pelo pipeline para reduzir a dominância das técnicas recorrentes; e (iv) incorporar métodos quantitativos capazes de estimar confiança e severidade das inferências produzidas. Além disso, pretende-se ampliar a cobertura da instrumentação para capturar evidências atualmente fora do escopo do pipeline, incluindo manipulação persistente de registro e carregamento lateral de bibliotecas. Também será investigado o uso de agentes automatizados para identificar comportamentos evasivos em tempo de execução e aplicar contramedidas dinamicamente durante a análise instrumentada, reduzindo o impacto de técnicas de anti-análise observadas no ambiente monitorado.

Declaração de Uso de Inteligência Artificial Generativa

Em conformidade com as diretrizes do SBSeg 2026 quanto à transparência no uso de tecnologias de Inteligência Artificial Generativa, os autores declaram a utilização da ferramenta *Composer 2.5 (Fast)* exclusivamente para apoio à revisão de estilo e clareza redacional do texto, incluindo correções gramaticais, ajustes de fluidez e uniformização terminológica e suporte auxiliar à implementação de códigos. A responsabilidade integral sobre o conteúdo do trabalho, incluindo a correção das afirmações técnicas e a integridade dos resultados reportados, permanece exclusivamente dos autores.

Agradecimentos

Os autores agradecem ao Programa de Pós-Graduação em Informática da Universidade Federal do Amazonas (PPGI/UFAM) e ao Instituto Federal do Amazonas (IFAM). Esta pesquisa foi parcialmente financiada, conforme previsto nos Arts. 21 e 22 do Decreto No. 10.521/2020, nos termos da Lei Federal No. 8.387/1991, através do convênio No. 015/2025, firmado entre ICOMP/UFAM, Digiboard da Amazônia Ltda e Lenovo Ltda. Esta pesquisa foi parcialmente financiada pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), por meio do PROEX – Código Financeiro 001, e pela Fundação de Amparo à Pesquisa do Estado do Amazonas (FAPEAM), Edital POSGRAD 2024/2025.

Referências

Apostolopoulos, T., Katos, V., Choo, K.-K. R., and Patsakis, C. (2021). Resurrecting anti-virtualization and anti-debugging: Unhooking your hooks. *Future Generation Computer Systems*, 116:393–405.

Balzarotti, D. et al. (2010). Efficient detection of split personalities in malware. In *NDSS*.

Botacin, M., Rocha, V. F. d., Geus, P. L. d., and Grégio, A. (2017). Analysis, anti-analysis, anti-anti-analysis: An overview of the evasive malware scenario. In *Anais do Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 250–263. SBC.

Bruening, D. et al. (2004). Efficient, transparent, and comprehensive runtime code manipulation. In *CGO*, pages 133–144. IEEE.

Campelo, H. B., Neto, F. S., and Feitosa, E. L. (2025). Contradef: Uma ferramenta de instrumentação binária dinâmica para análise de malware evasivo. In *Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg)*, pages 11–19. SBC.

Egele, M., Scholte, T., Kirda, E., and Kruegel, C. (2012). A survey on automated dynamic malware-analysis techniques and tools. *ACM Computing Surveys*, 44(2):1–42.

Frida Developers (2014). Frida: Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers. `https://frida.re`. Accessed: 2026-05-20.

Greamo, C. (2011). *Sandboxing*. Packt Publishing.

Issa, A. (2012). Anti-virtual machines and emulations. *Journal in Computer Virology*, 8(4):141–149.

Kirat, D. et al. (2011). Barebox: Efficient malware analysis on bare-metal. In *ACSAC*, pages 403–412. ACM.

Luk, C.-K. et al. (2005). Pin: Building customized program analysis tools with dynamic instrumentation. In *PLDI*, pages 190–200. ACM.

Nethercote, N. and Seward, J. (2007). Valgrind: A framework for heavyweight dynamic binary instrumentation. *ACM SIGPLAN Notices*, 42(6):89–100.

Neto, F. S., Campelo, H. B., Silva, E. V., and Feitosa, E. L. (2025). Mitigando técnicas de anti-instrumentação em dbi: Contramedidas baseadas em overhead e transparência. In *Simpósio Brasileiro de Cibersegurança (SBSeg)*, pages 1114–1121. SBC.

Or-Meir, O., Nissim, N., Elovici, Y., and Rokach, L. (2019). Dynamic malware analysis in the modern era—a state of the art survey. *ACM Computing Surveys*, 52(5):1–48.

Polino, M., Continella, A., Mariani, S., D'Alessio, S., Fontana, L., Gritti, F., and Zanero, S. (2017). Measuring and defeating anti-instrumentation-equipped malware. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 73–96. Springer.

Rodriguez, R. et al. (2016). Towards transparent and stealthy malware analysis systems using dbi. In *SAC*, pages 1964–1971. ACM.

Seifert, C., Steenson, R., Welch, I., Komisarczuk, P., and Endicott-Popovsky, B. (2007). Capture – a behavioral analysis tool for applications and documents. *Digital Investigation*, 4:S23–S30.

Shi, H. and Mirkovic, J. (2017). Hiding debuggers from malware with apate. In *Proceedings of the ACM Symposium on Applied Computing*, pages 1703–1710. ACM.

Shields, T. (2009). Anti-debugging – a developers view. *Proceedings of the 2009 Information Security Curriculum Development Conference*, pages 1–15.

Sun, K. et al. (2016). Break out of the truman sandbox: An emerging threat of native code to virtual machine introspection. In *CCS*, pages 969–982. ACM.

Ugarte-Pedrero, X. et al. (2019). A survey on malware analysis techniques. *IEEE Access*, 7:103783–103803.

Wu, Y. et al. (2023). A survey on dynamic binary instrumentation frameworks and their security applications. *ACM Computing Surveys*, 55(9):1–36.

You, I. and Yim, K. (2010). Malware obfuscation techniques: A brief survey. *2010 International Conference on Broadband, Wireless Computing, Communication and Applications*, pages 297–300.

Zhechev, M. (2018). Security analysis of dynamic binary instrumentation frameworks. Master's thesis, University of Twente.