RIP-ROP: uma proteção contra ataques de execução de código arbitrário baseados em *Return-Oriented Programming*[*]

Mateus F. Tymburibá Ferreira[1], Eduardo Luzeiro Feitosa[2]

[1]Autor – Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

[2]Orientador – Instituto de Computação – Universidade Federal do Amazonas – Manaus, AM – Brasil

mateustymbu@dcc.ufmg.br, efeitosa@icomp.ufam.edu.br

Abstract. *Return-Oriented Programming (ROP) is currently the main technique used by attackers to allow the execution of arbitrary code on vulnerable applications. While protections have been extensively studied, a definitive solution is still not known. This dissertation shows that ROP attacks can be prevented by controlling the frequency of indirect branch instructions executed by applications. Experiments with public exploits confirmed the feasibility of this model. This work also provides: (i) metrics for evaluation and comparison of protections against ROP attacks and (ii) taxonomies to classify those protections.*

Resumo. *Return-Oriented Programming (ROP) é atualmente a principal técnica usada por atacantes para permitir a execução de códigos arbitrários em aplicações vulneráveis. Apesar de proteções terem sido extensamente estudadas, ainda não se conhece uma solução definitiva. Esta dissertação demonstra que, através do controle da frequência de instruções de desvio indireto executadas pelas aplicações, é possível impedir ataques ROP. Experimentos realizados com exploits públicos confirmaram a viabilidade desse modelo. Este trabalho provê também: (i) métricas para avaliação e comparação de proteções contra ataques ROP e (ii) taxonomias para classificação dessas proteções.*

1. Introdução

Dados disponibilizados pelo NIST (*National Institute of Standards and Technology*) apontam para o crescimento do número de vulnerabilidades críticas identificadas em softwares [NIST 2013]. Entre elas, em decorrência do poder que proporcionam ao invasor, as vulnerabilidades que permitem a execução de código arbitrário conquistaram a preferência dos atacantes, conforme estatísticas catalogadas pelo CVE [CVE 2013].

---

* Apesar de atualmente o autor estar vinculado ao Departamento de Ciência da Computação da Universidade Federal de Minas Gerais, esta dissertação foi elaborada enquanto o autor era aluno de mestrado do Instituto de Computação da Universidade Federal do Amazonas. Sua versão completa está disponível em:

http://homepages.dcc.ufmg.br/~mateustymbu/Dissertacao_2014.pdf

1.1. Motivação

A técnica denominada *Return-Oriented Programming* (ROP) tem despertado grande interesse da comunidade científica e da indústria de segurança de sistemas, em função da sua larga utilização em ataques recentes de execução de código arbitrário. Entre muitos exemplos de códigos maliciosos de grande impacto que empregam ROP, pode-se citar os malwares Stuxnet e Duqu [Callas 2011].

Por ter se tornado o principal mecanismo utilizado por atacantes para desenvolver códigos maliciosos, mitigações contra ROP têm sido amplamente estudadas. O Windows 8, por exemplo, agrega um mecanismo de proteção contra ROP. No entanto, poucos dias depois do lançamento da versão preliminar do sistema, pesquisadores demonstraram estratégias relativamente simples capazes de burlar essa defesa [Son 2011]. Outro ponto de concentração de esforços na busca por mecanismos de contenção de ataques tem sido a ferramenta EMET (*Enhanced Mitigation Experience Toolkit*), que inclui uma defesa contra ROP premiada no concurso *BlueHat Prize*. Essa implementação foi superada duas semanas após o seu anúncio [Snake 2012]. De fato, diversas proteções contra ataques ROP já foram propostas, mas ainda não há uma solução efetiva.

1.2. Objetivos

Este trabalho tem dois objetivos: (i) introduzir métricas e taxonomias destinadas à avaliação de proteções contra ataques de execução de código arbitrário baseados em ROP e (ii) apresentar um método inédito de proteção contra esses ataques. A eficácia desse método no bloqueio de ataques reais é demonstrada através de testes com códigos maliciosos disponíveis em repositórios públicos de *exploits*. São analisadas também a ocorrência de falsos positivos e a eficiência computacional do modelo proposto, comparando-o com soluções correlatas.

1.3. Contribuições

Para atingir esses objetivos, as seguintes contribuições foram alcançadas:

- Definição de métricas para avaliação e comparação de proteções contra ataques ROP.
- Elaboração de duas taxonomias para classificação das proteções em função das estratégias de proteção contra ataques ROP e das abordagens de implementação utilizadas em cada solução.
- Demonstração da eficácia do controle da frequência de desvios indiretos como estratégia para detecção de ataques ROP.
- Desenvolvimento de um protótipo de proteção contra ataques ROP destinado a ambientes Windows e Linux em um *framework* de instrumentação binária dinâmica.

A estratégia de proteção baseada no controle da frequência de desvios indiretos, idealizada e validada neste trabalho, é suportada por elementos suficientes para sustentar a crença de que ela irá impactar na criação de melhores proteções contra

ataques ROP. Além disso, o protótipo desenvolvido, denominado RIP-ROP[1], viabiliza a proteção imediata de aplicações críticas executadas nos ambientes mencionados.

2. *Return-Oriented Programming*

Os ataques ROP foram criados com o intuito de superar a proteção oferecida pelo bit de execução (NX/XD), que impede a execução de instruções oriundas de áreas destinadas a dados, como a pilha de um processo [Shacham 2007]. *Exploits* ROP são baseados na capacidade de um atacante controlar, ao mesmo tempo, a pilha de execução de um programa e o conteúdo armazenado no registrador que indica o endereço da próxima instrução a ser executada. A partir dessas premissas, o atacante constrói uma entrada maliciosa que força o encadeamento de pequenos trechos de código da própria aplicação, conhecidos como *gadgets*.

Por limitações de espaço, este artigo não detalha a técnica ROP. No entanto, instruções detalhadas sobre o minucioso processo de construção de um *exploit* ROP podem ser encontradas na Seção 2.3 da dissertação e em um capítulo de livro publicado pelo autor deste trabalho no SBSeg 2012 (vide Seção 7.2). Esse capítulo de livro apresenta o conteúdo de um minicurso ministrado no evento e reflete parte dos ensinamentos colhidos durante a preparação para a elaboração da dissertação resumida neste artigo.

3. Trabalhos Relacionados

As estratégias de proteção contra ataques ROP são discutidas e classificadas segundo uma nova taxonomia (Seção 3.2). Além de classificar as principais estratégias, também introduzimos uma proposta de classificação segundo a abordagem utilizada pelas soluções para implementar as estratégias escolhidas (Seção 3.3). Essas inéditas taxonomias, propostas nesta dissertação, agrupam as proteções de acordo com características semelhantes, o que facilita a compreensão dos fatores que influenciam a qualidade dos esquemas de proteção.

3.1. Métricas de avaliação

As métricas propostas foram utilizadas durante a análise das características, virtudes e deficiências inerentes a cada um dos trabalhos avaliados. Da mesma forma, acredita-se que elas constituem valiosa ferramenta no processo de análise dos benefícios e limitações inerentes a novas proteções contra ataques ROP. As seguintes métricas foram consideradas relevantes:

- Tipos de ataques bloqueados: indica os tipos de *gadgets*[2] detectados pela proteção. Em ataques ROP, *gadgets* podem ser de três tipos: finalizados por instruções “Ret”, “Jump” indireto ou “Call” indireto [Checkoway et al. 2010].
- *Overhead* médio: indica a perda de desempenho que uma proteção impõem.
- Exceções: indica as situações em que códigos autênticos são classificados equivocadamente (falsos positivos).

1 http://homepages.dcc.ufmg.br/~mateustymbu/RIPROP.zip

2 *Gadgets*: pequenos segmentos de código finalizados por uma instrução de desvio indireto.

- Viabilidade prática do protótipo: indica se existe implementação da solução para proteção imediata de aplicações reais, em ambientes de produção.

3.2. Estratégias de Proteção

Observou-se que as proteções contra ROP têm utilizado estratégias semelhantes de detecção do ataque. Em função disso, propôs-se o agrupamento das soluções em classes, a fim de facilitar a compreensão de aspectos comuns às proteções pertencentes a um mesmo segmento. A taxonomia sugerida é composta pelos seguintes elementos: randomização, construção de uma pilha-sombra, checagem da instrução anterior ao endereço de retorno, checagem dos endereços autênticos para desvios, checagem da posição de entrada nas funções, controle da frequência de instruções de retorno e checagem do apontador para o topo da pilha. Em função do espaço limitado para resumir todo o conteúdo da dissertação, essas estratégias não serão detalhadas neste artigo. Todavia, cada uma dessas classes, bem como as proteções pertencentes a elas, são destrinchadas na Seção 3.2 da dissertação.

3.3. Abordagens de Implementação

As diversas estratégias de proteção, elencadas na Seção anterior, podem ser aplicadas aos sistemas que se pretende defender através de várias abordagens. Essas abordagens de implementação impactam diretamente na qualidade da solução proposta e apresentam características peculiares. Diante dessa percepção, foi proposta a seguinte taxonomia para classificação das proteções segundo as abordagens escolhidas para aplicar as medidas de prevenção contra ataques ROP: compilação, instrumentação binária estática, instrumentação binária dinâmica, adaptação do hardware, virtualização e emulação de código. Novamente, por falta de espaço, essas classes não são pormenorizadas neste artigo. Contudo, cada uma dessas abordagens é detalhada na Seção 3.3 da dissertação.

3.4. Discussão

A Tabela 3.1 da dissertação exibe um resumo da análise comparativa de 27 proteções contra ataques ROP à luz das métricas propostas e das taxonomias sugeridas. Dessa tabela, que não será reproduzida neste artigo por premência de espaço, depreende-se que os trabalhos que empregam a abordagem de Instrumentação Binária Dinâmica acarretam *overheads* maiores do que os trabalhos que utilizam as demais abordagens. Em contrapartida, as soluções que empregam Instrumentação Binária Estática e Adaptações do Hardware tendem a apresentar um *overhead* menor. Outra observação interessante reside no fato de que apenas 5 soluções parecem não apresentar problemas de compatibilidade com algum tipo de aplicação.

Tendo em vista que ainda não há uma proteção eficaz contra ataques ROP, espera-se que esse assunto continue atraindo a atenção de pesquisadores em busca de uma solução que alie eficácia no bloqueio de todas as variantes de ataques ROP, reduzido *overhead* computacional e baixo índice de ocorrência de exceções. A análise das proteções elaboradas até o presente momento permite, inclusive, que se formule afirmações mais gerais quanto ao futuro das proteções contra ataques ROP. Todas as soluções baseadas em verificações efetuadas em pontos específicos da execução de uma

aplicação, como as ocasiões em que chamadas de sistema são efetuadas, podem eventualmente ser superadas por estratégias que, após preparar o ataque, forcem a execução de códigos sem efeito simplesmente para iludir as checagens futuras. Por isso, acredita-se que soluções efetivas contra o ROP deverão checar o status do fluxo de execução durante toda a execução do processo.

4. Controle da frequência de desvios indiretos

Normalmente, as sequências de instruções que compõem cada *gadget* usado em um ataque ROP são extremamente curtas, dificilmente contendo mais do que cinco instruções. Essa é uma característica inerente aos ataques ROP, porque quanto maior a sequência de instruções, maior a probabilidade de existir entre essas instruções uma operação que altere o status da memória ou de um registrador de forma a comprometer o ataque. Essa alteração de status é comumente chamada por atacantes de "efeito colateral" de um *gadget*. A fim de evitar esses efeitos colaterais, quase sempre os *gadgets* escolhidos pelos atacantes são extremamente curtos.

Para evitar os mencionados "efeitos colaterais", ataques ROP apresentam uma elevada concentração de instruções de desvio indireto em um curto espaço de tempo. Diante dessa constatação, a estratégia de proteção proposta neste trabalho consiste em checar se a contagem do número de instruções de desvio indireto em uma determinada "janela de instruções" é maior do que um determinado limiar. Ao invés de medir a frequência apenas das instruções de retorno, esse novo esquema supervisiona a frequência de qualquer tipo de desvio indireto, incluindo aqueles efetuados através de instruções CALL indireto ou JMP indireto. Dessa forma, é possível evitar os três tipos de ataques ROP. Conforme pode ser observado na Tabela 3.1 da dissertação, poucas proteções oferecem esse benefício.

A fim de comprovar a eficácia dessa estratégia, elaborou-se uma metodologia que contemplou a análise do comportamento dos desvios indiretos tanto em aplicações normais quanto em ataques ROP. Através dessa avaliação, foi possível confirmar que a diferença de comportamento entre essas duas classes é suficiente para ratificar a hipótese de que o controle da frequência de desvios indiretos é uma estratégia eficaz na detecção de ataques ROP (Seção 5.3 da dissertação). A análise da frequência de desvios indiretos foi dividida em duas etapas:

- Avaliação teórica (Seção 4.1.2 da dissertação): foram estudados os cenários em que o comportamento de aplicações autênticas mais se aproxima do padrão apresentado por ataques ROP.
- Avaliação empírica (Seção 4.1.3 da dissertação): foram registrados os valores de frequência máxima de desvios indiretos observados durante a execução dos *benchmarks* que compõem a suíte SPEC CPU2006, comparando-os com as frequências máximas de desvios indiretos observadas durante a execução de 15 *exploits* ROP reais. Os valores ideais do tamanho da janela de instruções, bem como do limiar, também foram definidos através desses experimentos.

Além de demonstrar a eficácia da proteção no bloqueio a ataques ROP, foram executados procedimentos para a avaliação do impacto da solução no tempo de execução das aplicações protegidas. Para isso, comparou-se o desempenho dos

benchmarks que compõem o SPEC CPU2006 com o tempo de CPU observado quando esses mesmos aplicativos são protegidos pelo RIP-ROP e por soluções correlatas. Os detalhes de implementação e as otimizações agregadas ao protótipo estão descritas na Seção 4.2 da dissertação e não serão apresentados neste artigo por restrição de espaço.

5. Resultados

Ao analisar os resultados dos experimentos realizados (Capítulo 5 da dissertação), constata-se que a densidade máxima de instruções de desvio indireto é maior nos *exploits* ROP do que em aplicações autênticas. Ao mesmo tempo, não foi identificado qualquer caso de falso positivo entre todos os *benchmarks* e ambientes de experimentação. Além disso, o RIP-ROP foi testado com sucesso na proteção de 15 (quinze) aplicações para as quais existem *exploits* ROP publicamente disponíveis.

A Tabela 1 exibe uma comparação do RIP-ROP com outras proteções que utilizam instrumentadores binários dinâmicos para implementar soluções baseadas no controle da frequência de instruções. Nessa tabela, estão expressos os *overheads* reportados pelos autores de cada solução, que remontam a diferentes conjuntos de teste e ambientes de experimentação. Apesar disso, pode-se dizer que o custo computacional apresentado pelo RIP-ROP (727%) é comparável ao do DROP [Chen et al. 2009] (530%), que utiliza o mesmo *framework* de instrumentação binária, mas mediu o desempenho através da execução de uma seleção de aplicações, ao invés da suíte de *benchmarks* SPEC. As únicas aplicações utilizadas tanto nos testes executados com o DROP quanto nos experimentos realizados com o RIP-ROP (bzip2 e gcc), que podem oferecer uma comparação mais realista, indicam que o protótipo desenvolvido neste trabalho impõe um *overhead* menor. Nos experimentos com o bzip2, o DROP acarretou em um custo computacional de 1.540%, consideravelmente superior aos 809% registrados pelo RIP-ROP. Nos testes com o gcc, o DROP impôs um *overhead* de 960%, enquanto o RIP-ROP elevou o tempo de CPU em 729%.

Tabela 1: Proteções que controlam a frequência de instruções de retorno via instrumentação binária dinâmica

Proteção	*Abordagens*	*Estratégias*	*Ataques bloqueados*	*Overhead (%)*	*Exceções*	*Viabilidade prática*
DynIMA [Davi et al. 2009]	2 e 3	6	R	*	4	Não
DROP [Chen et al. 2009]	3	6	R	530,0	1	Sim
RIP-ROP	**3**	**8**	**R, J e C**	**727**	**0**	**Sim**

Abordagens: 2-Instrumentação Binária Estática ; 3-Instrumentação Binária Dinâmica.
Estratégias: 6-Controle da frequência de instruções de retorno ; 8-Outras.
Ataques bloqueados: R-encadeamento via RET; J-encadeamento via JMP; C-encadeamento via CALL.
*O *overhead* não é informado na publicação.

Outro fator de comparação entre as proteções, ilustrado na Tabela 1, recai sobre os tipos de ataques ROP bloqueados. Nesse caso, apenas o RIP-ROP oferece uma proteção contra todos os tipos de *exploits* ROP. Essa capacidade está diretamente relacionada à mudança na estratégia de detecção dos ataques adotada neste projeto, que amplia o escopo de monitoramento para abarcar todas as instruções de desvio indireto.

6. Conclusão

Levando-se em conta a grande quantidade de proteções contra ataques ROP publicadas

e a complexidade do processo de comparação dessas soluções, este trabalho contribui com a pesquisa nessa área ao apresentar uma visão geral dos trabalhos, estruturar as soluções em duas classificações e propor métricas para a análise e comparação de proteções. Além disso, este trabalho demonstrou que a imposição de um limite para o uso de instruções de desvio indireto acarreta em severas limitações à capacidade de criação de um *exploit* ROP efetivo, impossibilitando-a em todos os casos testados. Finalmente, também foi desenvolvido neste trabalho um protótipo que pode ser facilmente adotado em ambientes de produção. A análise de desempenho desse protótipo indicou que a proteção é obtida a um custo computacional comparável à performance de soluções correlatas, superando-as em alguns casos.

7. Produção científica

Esta Seção relaciona a produção científica resultante da dissertação.

7.1. Trabalhos premiados

- Tymburibá, Mateus, Moreira, Rubens e Pereira, Fernando (2016). Inference of peak density of indirect branches to detect ROP attacks. *Proceedings of the ACM International Symposium on Code Generation and Optimization*, p. 150-159. http://dl.acm.org/citation.cfm?id=2854049

Esse artigo apresenta uma extensão do trabalho desenvolvido na dissertação. Ele descreve um algoritmo capaz de inferir o limiar máximo de frequência de instruções de desvio indireto previsto para uma aplicação à partir do seu código-fonte. Recebeu o prêmio de melhor trabalho de pesquisa de estudante (*ACM Student Research Competition*) em um dos principais eventos internacionais para divulgação de resultados de pesquisas ligadas à análise e otimização de códigos (CGO 2016), evento com classificação A2 no sistema Qualis da CAPES.

- Tymburibá Ferreira, M., Filho, A. e Feitosa, E. (2014). Controlando a Frequência de Desvios Indiretos para Bloquear Ataques ROP. *Anais do XIV SBSeg*, p. 223–236. http://www.lbd.dcc.ufmg.br/colecoes/sbseg/2014/0017.pdf

Esse artigo discute parte dos resultados apresentados na dissertação. Recebeu menção honrosa para melhor artigo no SBSeg 2014, evento com classificação B4 no sistema Qualis da CAPES.

7.2. Outros trabalhos

- Emilio, R., Tymburibá, M. ; Pereira, F. (2015). Inferência Estática da Frequência Máxima de Instruções de Retorno para Detecção de Ataques ROP. *Anais do XV SBSeg*, p. 2–15. http://sbseg2015.univali.br/anais/SBSegCompletos/artigoCompleto01.pdf

Esse artigo apresenta uma versão preliminar do algoritmo publicado em CGO 2016. Foi publicado no SBSeg 2015, evento com classificação B4 no sistema Qualis da CAPES.

- Tymburibá, M., Moreira, R. e Pereira, F. (2015). RipRop: A Dynamic Detector of ROP Attacks. *Anais da Sessão de Ferramentas do Congresso Brasileiro de Software*. p. 9–16. http://cbsoft.org/articles/0000/0529/Ferramentas.pdf

Esse artigo descreve as otimizações e os detalhes de implementação do RIP-ROP. Foi publicado no CBSoft 2015, congresso sem classificação no sistema Qualis da CAPES que agrega 4 eventos tradicionais: SBES (Qualis B2), SBLP (Qualis B3), SBMF (Qualis B4) e SBCARS (Qualis B4).

- Tymburibá, M. (2015). Counting the Frequency of Indirect Branches to Detect Return-Oriented Programming Attacks. *Student Forum Supplementary Volume of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN).* http://homepages.dcc.ufmg.br/~mateustymbu/DSN-2015.pdf

Esse artigo descreve uma solução em hardware baseada na estratégia de proteção desenvolvida nesta dissertação. Foi apresentado no fórum de estudantes da 45ª *IEEE/IFIP International Conference on Dependable Systems and Networks* (DSN), uma das conferências internacionais de maior prestígio para apresentar trabalhos na área de segurança da computação, com classificação A1 no sistema Qualis da CAPES.

- Tymburibá Ferreira, M., Rocha, T. de S., Martins, G. B., Feitosa, E. e Souto, E. (2012). Análise de vulnerabilidades em Sistemas Computacionais Modernos: Conceitos, Exploits e Proteções. *Minicursos do XII SBSeg*. p. 2–51. http://sbseg2012.ppgia.pucpr.br/@docs/SBSeg2012Minicursos.pdf

Esse capítulo de livro descreve as principais técnicas modernas de desenvolvimento de *exploits*, incluindo ROP. Foi publicado e apresentado no formato de minicurso no SBSeg 2012, evento com classificação B4 no sistema Qualis da CAPES.

Referências

Callas, J. (2011). Smelling a RAT on Duqu. http://blogs.entrust.com/enterprise-authentication/?p=236.

Checkoway, S., Davi, L., Dmitrienko, A., et al. (2010). Return-oriented programming without returns. *CCS*, p. 559–572.

Chen, P., Xiao, H., Shen, X., et al. (15 nov 2009). DROP: Detecting Return-Oriented Programming Malicious Code. *ICISS*, Lecture Notes in Computer Science. v. 5905, p. 163–177.

CVE (2013). Vulnerability distribution of cve security vulnerabilities by types. http://www.cvedetails.com/vulnerabilities-by-types.php.

Davi, L., Sadeghi, A. e Winandy, M. (2009). Dynamic integrity measurement and attestation: towards defense against return-oriented programming attacks. In *Proceedings of the 2009 ACM workshop on Scalable trusted computing*.

NIST (2013). National Vulnerability Database (NVD) CVE Statistics. https://web.nvd.nist.gov/view/vuln/statistics-results?adv_search=true&cves=on&cwe_id=CWE-119.

Shacham, H. (2007). The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86). *CCS*,

Snake (2012). Bypassing EMET 3.5's ROP Mitigations. https://repret.wordpress.com/2012/08/08/bypassing-emet-3-5s-rop-mitigations/.

Son, N. H. (2011). ROP chain for Windows 8. http://blog.bkav.com/en/rop-chain-for-windows-8/.

