Forseti: Extração de características e classificação de binários ELF

Lucas Galante[1], **Marcus Botacin**[2], **André Grégio**[2], **Paulo Lício de Geus**[1]

[1] Universidade Estadual de Campinas (UNICAMP)

{galante, paulo}@lasca.ic.unicamp.br

[2]Universidade Federal do Paraná (UFPR)

{mfbotacin, gregio}@inf.ufpr.br

Abstract. *Malware infections are constant threats to multiple computing platforms and binary classification leveraging machine learning (ML) techniques has been demonstrated to be a promising approach for fighting these infections. Currently, most ML solutions focus only on the* `Windows` *platform. To bridge this development gap, we present* `Forseti`*, a solution for feature extraction and classification of* `Linux` *ELF binaries.*

Resumo. *A infecção por códigos maliciosos é uma ameaça constante a múltiplas plataformas computacionais e a classificação de binários utilizando-se de técnicas de aprendizado de máquina tem se mostrado como uma promissora forma de combater tais infecções. Atualmente, a maioria das soluções deste tipo é focada somente na plataforma* `Windows`*. De modo a suprir esta lacuna de desenvolvimento, apresentamos* `Forseti`*, uma solução de extração de características e classificação de binários ELF para a plataforma* `Linux`*.*

1. Classificadores de *Malware*

Programas maliciosos (*malware*) são uma preocupação constante para empresas e usuários domésticos em virtude dos extensos danos causados aos sistemas computacionais atuais, tais como prejuízos financeiros decorrentes do sequestro digital de dados, prática implementada por códigos maliciosos conhecidos como *ransomware* [BBC 2017]. Esta preocupação deu origem a uma frequente busca por soluções de detecções de binários maliciosos que possibilitem a prevenção de infecções e/ou o bloqueio de ameças antes que estas causem danos extensos aos sistemas infectados.

Atualmente, as abordagens mais promissoras para a detecção de códigos maliciosos se baseiam em aprendizado de máquina [Babaagba and Adesanya 2019, Kruczkowski and Szynkiewicz 2014]. Este tipo de solução se caracteriza por extrair características (e.g., identificadores) dos binários suspeitos e compará-las com um conjunto de dados conhecidos, atribuindo ao binário um rótulo de suspeito ou não, de acordo com a proximidade das características do binário suspeito em comparação com a dos binários conhecidos. Este procedimento é ilustrado na Figura 1.

Abordagens de classificação de binários utilizando-se de aprendizado de máquina evoluíram de técnicas focadas apenas em maximizar a acurácia para técnicas focadas também nas características dos dados [Feizollah et al. 2015, Ahmadi et al. 2016], uma preocupação também deste trabalho. Apesar desta evolução, poucas soluções são focadas

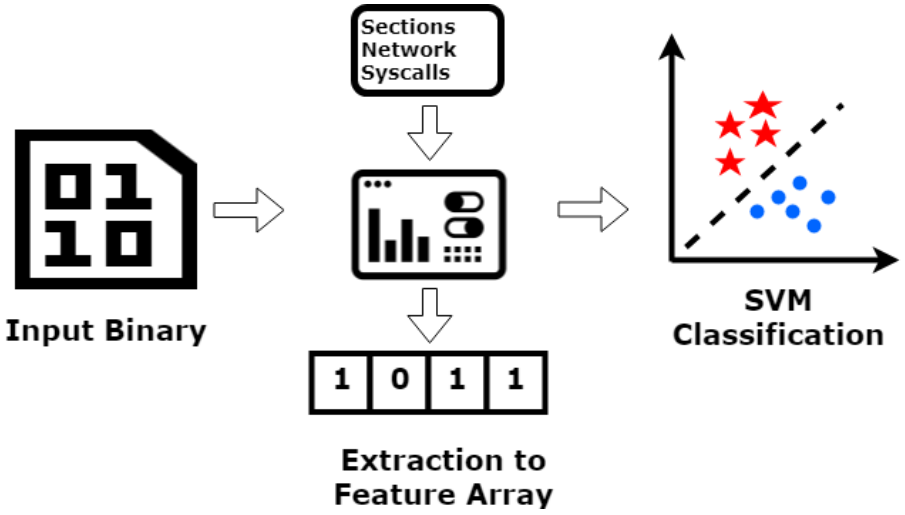

Figura 1. Arquitetura abstrata dos processos de extração de características e classificação de binários utilizando técnicas de aprendizado de máquina.

em binários `ELF` (*Executable and Linkable Format*) [O'Neill 2016] para a plataforma `Linux`, uma lacuna de desenvolvimento considerada neste trabalho. Desta forma, propomos `Forseti`, uma solução de extração de características de binários e classificação de binários `ELF`.

Para o desenvolvimento do `Forseti`, desenvolvemos e nos baseamos em estudos prévios sobre as características de códigos maliciosos na plataforma `Linux` [Galante et al. 2018], de modo a prever extratores de características direcionados para as ameaças presentes nesta plataforma.

2. `Forseti`: Arquitetura & Implementação

Nesta seção, apresentamos a arquitetura proposta e a implementação do `Forseti`.

2.1. Arquitetura

`Forseti` é uma solução autocontida para extração de características e classificação de binários. Portanto, dado um conjunto de binários de entrada, divididos entre (i) binários maliciosos rotulados, (ii) binários benignos rotulados e (iii) binários não rotulados, `Forseti` automaticamente realiza a extração de características de todos os binários, realiza o treino dos algoritmos de aprendizado de máquina a partir dos dados rotulados e prediz a classe (malicioso ou benigno) dos binários não rotulados. Internamente, `Forseti` é implementado por duas classes de componentes: (i) extratores de características; e (ii) diversos classificadores; como mostrado na Figura 2.

2.2. Implementação

`Forseti` é inteiramente implementado em `Python`, fazendo uso de bibliotecas. A extração de características de binários `ELF` é realizada pela biblioteca `pyelftools` [Eliben 2019]. A classificação de binários é realizada pela biblioteca `sklearn` [Pedregosa et al. 2011].

`Forseti` é uma solução altamente flexível, permitindo que o utilizador (analista) selecione as características dos binários que deseja considerar em suas análises. A seleção é realizada através de arquivos de configuração (*conf files*) tratados pela biblioteca

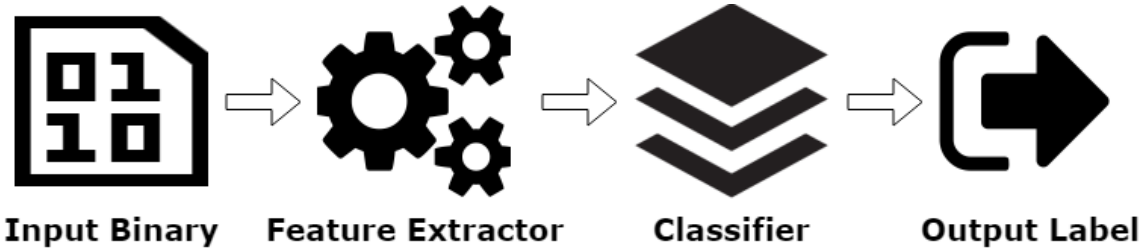

Figura 2. Arquitetura do `Forseti`. Etapas sequenciais são implementadas por módulos independentes.

`ConfigParser` [ConfigParser 2019]. Além disso, `Forseti` permite ao analista decidir se a predição de um binário não rotulado deve se dar com uma base já treinada ou a partir de um novo treinamento. `Forseti` armazena e recupera treinamentos através da serialização dos objetos dos classificadores pela biblioteca `pickle` [Pickle 2019].

Através do carregamento dos classificadores previamente treinados, analistas fazendo uso do `Forseti` podem classificar binários desconhecidos utilizando-se de diferentes conjuntos de características, de acordo com as necessidades da tarefa. Analistas podem também armazenar os classificadores treinados para posterior reprodução dos resultados.

2.2.1. Características Extraídas

`Forseti` é capaz de extrair diferentes características dos binários `ELF`, conforme seleção realizada pelo utilizador. Os extratores implementador por `Forseti` são apresentados na Tabela 1. Para maiores detalhes sobre a extração de características na plataforma `Linux`, recomendamos a leitura da litetura específica sobre o tema [Botacin et al. 2019b].

Tabela 1. Características dos binários de acordo com método de extração (estático ou dinâmico) e representação (discreta ou contínua).

Característica	Método de Extração	Representação	Descrição
`fork` *syscall*	Ambas	Ambas	Chamada da *syscall* `fork`
`ptrace` *syscall*	Ambas	Ambas	Chamada da *syscall* `ptrace`
`socket` *syscall*	Dinâmico	Ambas	Chamada da *syscall* `socket`
`mmap` *syscall*	Dinâmico	Ambas	Chamada da *syscall* `mmap`
sinal `SIGTERM`	Dinâmico	Ambas	Término via `SIGTERM`
sinal `SIGSEGV`	Dinâmico	Ambas	Término via `SIGSEGV`
acesso `/proc`	Ambas	Ambas	Acesso ao diretório `/proc`
acesso `/home`	Ambas	Ambas	Acesso ao diretório `/home`
acesso `passwd`	Ambas	Ambas	Acesso ao arquivo `passwd`
permission denied	Dinâmico	Ambas	Permissão negada ao acessar recurso
Tamanho seções	Estático	Contínua	Tamanho das seções do binário
Tamanho exemplar	Estático	Contínua	Tamanho (em *bytes*) do exemplar
Arquivos embutidos	Estático	Discreta	Binário contém arquivos embutidos
Ligação (Tipo)	Estático	Discreta	Tipo de ligação (estática ou dinâmica)
bibliotecas (#)	Estático	Contínua	Quantidade de bibliotecas dinâmicas
`UPX`	Estático	Discreta	Binário está empacotado por `UPX`
cabeçalhos (#)	Estático	Contínua	Número de *headers* do binário
.dynamic (#)	Estático	Contínua	Número de seções de tipo dinâmico
seções (#)	Estático	Contínua	Número de seções do binário
Falha *disassembly*	Estático	Discreta	Falha no *disassembly* via `objdump`

2.2.2. Classificadores Implementados

`Forseti` implementa diferentes classificadores que podem ser utilizados, em conjunto ou individualmente, pelo analista para identificar binários não rotulados. Atualmente, os seguintes classificadores podem ser utilizados:

- ***Support Vector Machines* (SVM)** [Wang 2005]: Método de classificação utilizando-se de um hiperplano para a separação dos vetores de características e atribuição de rótulos.
- ***Random Forest* (RF)** [Breiman 2001]: Método de classificação utilizando-se de árvores de decisão para a atribuição de rótulos.
- ***Multi Layer Perceptron* (MLP)** [Rathbun 1997]: Método de classificação utilizando-se de redes neurais de múltiplas camadas e retro-propagação de erro para a predição de rótulos.

3. Experimentação

Nesta seção, descrevemos como `Forseti` pode ser obtido e utilizado.

Download. `Forseti` é uma solução de código aberto e pode ser baixado em: `https://github.com/marcusbotacin/ELF.Classifier`.

Instalação. `Forseti` pode ser usado a partir da instalação de dependências: *python 2*, *sklearn*, *pyelftools* e *pickle*.

Execução. A operação do `Forseti` ocorre através da linha de comando e arquivo de configuração. *python main.py -h* exibe os argumentos para determinação dos binários e.g., *python main.py -m malware-list -g goodware-list*. O arquivo *forseti.conf* contém configurações de operação da ferramenta, como a definição do classificador e do modo de operação.

Os principais comandos para a operação do `Forseti` são:

- **Treino:**
  - Arquivo de configuração: `Validation`
  - Comando: `python main.py -m malware -g goodware`
- **Predict:**
  - Arquivo de configuração: `Testing`
  - Comando: `python main.py -m malware -g goodware -s unknown`

4. Estudos suportados por `Forseti`

`Forseti` tem sido utilizado como suporte para o desenvolvimento de pesquisas acadêmicas com foco na classificação de binários `ELF`. Destacamos, como exemplo, o uso de `Forseti` na avaliação de classificadores e modelos de classificação [Galante et al. 2019] e como *groundtruth* para a implementação de classificadores em *hardware* [Botacin et al. 2019a].

Agradecimentos

Marcus Botacin é financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, Bolsa de Doutorado, processo 164745/2017-3). Lucas Galante é financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, Bolsa PIBIC). Todos os autores são financiados pela Coordenação para o Aperfeiçoamento Pessoal do Ensino Superior (CAPES, Projeto FORTE, Programa de Ciências Forenses, 24/2014, processo 23038.007604/2014-69).

Referências

Ahmadi, M., Ulyanov, D., Semenov, S., Trofimov, M., and Giacinto, G. (2016). Novel feature extraction, selection and fusion for effective malware family classification. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, CODASPY '16, pages 183–194, New York, NY, USA. ACM.

Babaagba, K. O. and Adesanya, S. O. (2019). A study on the effect of feature selection on malware analysis using machine learning. In *Proceedings of the 2019 8th International Conference on Educational and Information Technology*, ICEIT 2019, pages 51–55, New York, NY, USA. ACM.

BBC (2017). Ransomware attacks around the world grow by 50 `http://www.bbc.com/news/technology-39730407`.

Botacin, M., Galante, L., Ceschin, F., Santos, P., Carro, L., de Geus, P., Grégio, A., and Zanata, M. (2019a). The av says: Your hardware definitions were updated! *14th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC 2019)*.

Botacin, M., Galante, L., Silva, O., and de Geus, P. (2019b). Introdução à engenharia reversa de aplicações maliciosas em ambientes linux. *Minicursos do XIX SBSEG*.

Breiman, L. (2001). Random forests. *Mach. Learn.*, 45(1):5–32.

ConfigParser (2019). Configuration file parser. `https://docs.python.org/2/library/configparser.html`.

Eliben (2019). Pyelftools. `https://github.com/eliben/pyelftools`.

Feizollah, A., Anuar, N. B., Salleh, R., and Wahab, A. W. A. (2015). A review on feature selection in mobile malware detection. *Digit. Investig.*, 13(C):22–37.

Galante, L., Botacin, M., Grégio, A., and de Geus, P. (2018). Malicious linux binaries: A landscape. *Workshop de Trabalhos de Iniciação Científica e Conclusão de Curso de Gradução do XVIII SBSEG*.

Galante, L., Botacin, M., Grégio, A., and de Geus, P. (2019). Machine learning for malware detection: Beyond accuracy rates. *Workshop de Trabalhos de Iniciação Científica e Conclusão de Curso de Gradução do XIX SBSEG*.

Kruczkowski, M. and Szynkiewicz, E. N. (2014). Support vector machine for malware analysis and classification. In *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 02*, WI-IAT '14, pages 415–420, Washington, DC, USA. IEEE Computer Society.

O'Neill, R. E. (2016). *Learning Linux Binary Analysis*. Packt Publishing.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pickle (2019). Pickle. `https://docs.python.org/2/library/pickle.html`.

Rathbun, T. F. (1997). *Autonomous Construction of Multilayer Perceptron Neural Networks*. PhD thesis, Air Force Institute of Technology, Wright Patterson AFB, OH, USA. UMI Order No. GAX97-32738.

Wang, L. (2005). *Support Vector Machines: Theory and Applications (Studies in Fuzziness and Soft Computing)*. Springer-Verlag, Berlin, Heidelberg.