Simulação de *Penny Attack* no Ethereum e sua Identificação usando Classificadores

José Eduardo de A. Sousa[1], Vinicius C. Oliveira[1], Júlia Valadares[1], Alex B. Vieira[1], Heder S. Bernardino[1], Saulo M. Villela[1], Glauber Dias[2]

[1]Departamento de Ciência da Computação
Universidade Federal de Juiz de Fora (UFJF), Juiz de Fora – MG, Brasil

[2]Campus Senador Helvído Nunes de Barros
Universidade Federal do Piauí (UFPI), Picos – PI, Brasil

{eduardo2,vinicius.oliveira,juliavaladares,heder}@ice.ufjf.br,

{alex.borges,saulo.moraes}@ufjf.edu.br, ggoncalves@ufpi.edu.br

Abstract. *The growing interest in Ethereum leads to concerns in its security, given that there have already been attacks that exploited its charging mechanism or Penny Attack. These attacks affected the network, causing transactions congestion and there are indications that Ethereum remains susceptible to this type of attack. We analyze the behavior of the network during a Penny Attack, assuming its negligible cost to the attacker and seeking machine learning techniques to detect it previously, using transaction attributes. Our techniques had an AUC, F_β and recall greater than 94%, 82% and 98% respectively.*

Resumo. *O crescimento do interesse em Ethereum leva a preocupações relacionadas à sua segurança, dado que já houveram ataques que exploraram o seu mecanismo de tarifação ou Penny Attack. Esses ataques afetaram a rede ocasionando lentidão nas transações e há indícios que Ethereum continua susceptível a esse tipo de ataque. Analisamos o comportamento da rede Ethereum durante um Penny Attack, buscando técnicas de aprendizado de máquina para detectá-lo previamente, utilizando atributos das transações. Nossas técnicas tiveram AUC, F_β e recall superior a 94%, 82% e 98% respectivamente.*

1. Introdução

O crescimento do interesse em Ethereum leva a preocupações relacionadas à segurança desse sistema, dado o seu potencial de uso em aplicações financeiras. Dentre as várias possibilidades de ataques, ou falhas de segurança que sistemas de criptomoedas estão propensos, os ataques distribuídos de negação de serviço (*Distributed Denial of Service* - DDoS) estão entre os maiores desafios a serem enfrentados [Chen et al. 2017]. Nesse tipo de ataque várias transações maliciosas são efetuadas em um curto período, mantendo o sistema ocupado, de modo que não possa atender totalmente as transações reais.

Mesmo que as transações financeiras sejam realizadas mediante o pagamento de uma tarifa, como ocorre na maioria das criptomoedas, usuários maliciosos têm explorado esse esquema de tarifação para aplicar ataques DDoS. Por exemplo, a plataforma Ethereum, em especial, já teve de se adequar a ataques que exploravam o seu mecanismo de

tarifação para tornar todas as transações da rede mais lentas, ocasionando desperdício de recursos computacionais [Gautham 2016, Buterin 2016].

Neste trabalho, temos o objetivo de analisar o comportamento da rede Ethereum em momentos de ataque, aqui denominados como *Penny Attack* [Gerard 2017]. Para investigar essa questão, primeiramente assumimos que o custo do *Penny Attack* é negligenciável para atacantes que visam desestabilizar a rede Ethereum. A seguir, estabelecemos uma definição de transações maliciosas no Ethereum e, desenvolvemos um ambiente experimental controlado que busca replicar a rede Ethereum atual com a inserção de transações maliciosas, analisando comportamento da rede. Como resultados de nossas análises propomos uma técnica para detecção baseado em características das transações e aprendizado de máquina.

A detecção de *Penny Attack* pode diminuir os seus efeitos nocivos à rede. Em particular, a aplicação do aprendizado de máquina ganhou notoriedade devido ao seu poder de identificar antecipadamente comportamentos previamente conhecidos de maneira ágil e confiável. Essa agilidade é importante na detecção dos ataques para reduzir os danos causados pelos usuários maliciosos. Portanto, propomos o uso de um modelo de previsão para identificar *Penny Attack*. Especificamente, técnicas de aprendizado de máquina, incluindo comitês de classificadores, foram aplicados com o objetivo de reconhecer padrões de transações maliciosas na rede. Os resultados de nossas avaliações mostram que esses modelos foram capazes de identificar *Penny Attack*.

2. Metodologia

Definimos um ambiente de experimentação controlado para avaliar o comportamento da rede Ethereum com *Penny Attack* através de uma simulação. A Figura 1 ilustra a arquitetura geral que projetamos para a simulação da rede ETH. Essa arquitetura está organizada em quatro componentes: *(1) base de dados, (2) inicialização da rede, (3) controle e (4) comunicação.* Esses componentes se comunicam para execução de transações na rede ETH. As setas indicam a dinâmica de funcionamento da arquitetura, dado o fluxo de dados entre os componentes e a rede.

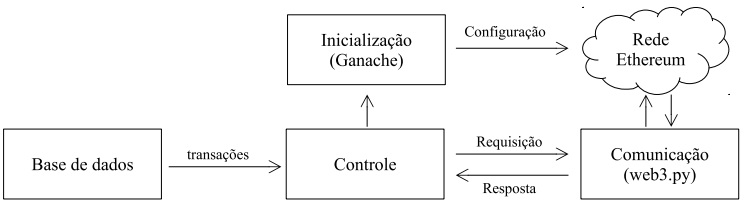


Figura 1. Visão geral da arquitetura para executar os experimentos.

A primeira etapa da simulação ocorre com o componente base de dados, que fornece as transações a serem processadas nessa rede. Essas transações podem ser reais extraídas do Ethereum ou outras criptomoedas, assim como transações sintéticas em que características como tarifa, valor, origem e destino são definidas de forma personalizadas. Utilizamos também transações sintéticas como forma de produzir comportamento condizente a um ataque à rede. Nesse caso, configuramos essas transações com valores encontrados nas transações reais para evitar previsibilidade de comportamento malicioso

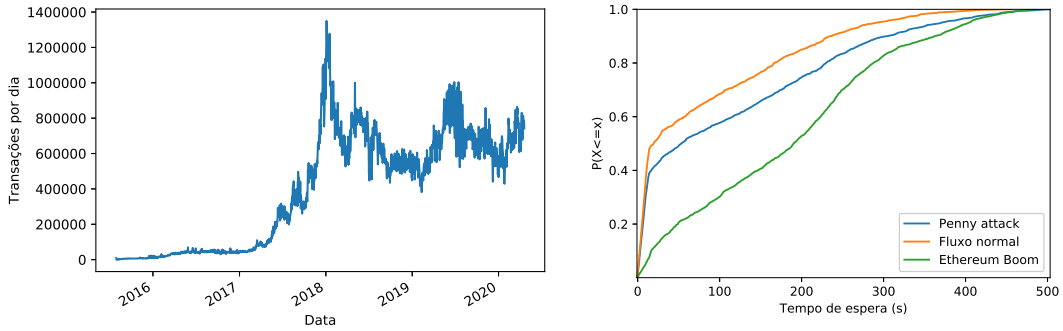
modificando apenas o preço do gas para minimizar o custo do ataque em termos de tarifa. Especificamente, definimos o preço de gas como 1 GWei (10^{-9} Ether), que equivale ao 17o. percentil dos valores de preço do gas que observamos nas transações reais.

Dado as transações, a próxima etapa da simulação ocorre no componente inicialização que utiliza a ferramenta Ganache. O Ganache é responsável por prover o serviço do Ethereum, dispondo de um facilitador de configuração da rede. Assim, configuramos uma rede com tempo entre blocos de 13 segundos e um número de contas com saldo na rede igual a quantidade de endereços únicos contidos na base de transações utilizadas. Dado essas configurações, o controle dispara um comando para que o Ganache gere uma nova instância de rede Ethereum privada. Como pode ser observado na Figura 1, o componente controle é central na arquitetura, e nos permite monitorar todas as etapas das simulações. Em particular, o controle faz o mapeamento de usuários extraídos das transações reais em usuários da rede inicializada na simulação, além de definir uma frequência de injeção de transações por período, monitoramento da fila de transações (mempool) e a solicitação da criação de novas transações e contratos inteligentes. O componente comunicação, por sua vez, utiliza a API *web3.py* para requisitar a execução de transações à rede, assim como fornecer a resposta dessas execuções.

Dado a visão geral da arquitetura, descrevemos agora o comportamento configurado no controlador para as simulações. Executamos a simulação em um período de 15 minutos da rede para simular o ataque. A injeção de carga na rede foi dívida em três partes de 5 minutos. A primeira parte consiste em injetar dados na rede acima da média de transações por segundo para obter, além de transações suficientes para estabelecer um bloco, transações que podem compor a fila de transações pendentes. Isso equivale a 16,7% da taxa de chegada das transações reais. Em uma segunda etapa, as transações de comportamento normal serão inseridas simultaneamente com as transações de comportamento malicioso geradas artificialmente. Em um terceiro momento, a rede terá apenas a inserção de transações da rede principal em fluxo proporcional ao convencional, para verificar como a rede se comporta após o ataque.

2.1. Aprendizado de Máquina

Neste trabalho utilizaremos os classificadores Árvore de Decisão, *Random Forest*, KNN, SVM e Naïve Bayes, além de um conjunto de classificadores, definindo a classificação não apenas com base em um único algoritmo mas com base na decisão conjunta de um comitê . Essa abordagem permite que as vantagens de diversos tipos de classificadores sejam combinadas, buscando eliminar seus pontos fracos e, em seguida, levar a uma solução mais robusta. As predições para cada instância é feita com base num critério de votação definido ao criar este comitê, o qual pode ser definido como: (i) voto majoritário (*hard*) e (ii) média de confiança (*soft*). O primeiro modo parte do princípio simples de que a classe de resposta mais votada pelos modelos participantes do comitê será a classe escolhida como resposta. A segunda opção tem um funcionamento diferente, partindo sua decisão da certeza que os modelos têm de que uma transação é pertencente a uma classe e, assim, apresenta uma resposta com base numa média.



(a) Transações por dia no Ethereum (ETH).

(b) Função de distribuição acumulada do tempo de espera da simulação.

Figura 2. Dados mensurados da rede ETH e de simulação.

3. Experimentos Computacionais

Utilizamos dados de transações da rede ETH que foram apresentados em nossos trabalhos anteriores [Sousa et al. 2019, Sousa et al. 2020]. que utilizam o Etherscan[1]. A Figura 2(a) mostra um gráfico com a quantidade de transações que trafegam na rede por dia. Os atributos mostrados na Tabela 1 são os adotados aqui, que foram suficientes para reproduzirmos o comportamento da rede ETH em nossos experimentos. É importante observar que o atributo *tempo de espera* foi calculado a partir de outros atributos disponíveis. Com a simulação, calculamos o momento de criação da transação e o momento em que um bloco foi inserido contendo esta transação. A diferença entre esses valores é referida aqui como tempo de espera.

Tabela 1. Atributos obtidos de transações no Ethereum.

Atributo	Descrição	Calculado?
hash	Texto único de tamanho fixo usando todos dados da transação.	Não
origem	Remetente da transação.	Não
destino	Destinatário da transação.	Não
valor	Quantidade de Ether transferido do remetente ao destinatário.	Não
gas	Unidade de esforço computacional necessário para executar uma transação com sucesso.	Não
limite do gas	Unidade de esforço computacional oferecido para executar uma transação.	Não
preço do gas	Quantidade (na moeda Ether) paga por gas.	Não
tempo de espera	Tempo entre criação da transação e quando aparece na blockchain.	Sim

Para a simulação, especificamente, usamos transações que aparecem na rede no dia 16/05/2019 entre 08:22 e 08:37, onde são replicados seus valores de gas, preço do gas e valor transacionado. Reproduzimos três cenários em nossas simulações da rede: (i) um fluxo normal com 2000 transações Ethereum, (ii) *Penny Attack* com as mesmas transações do fluxo normal, mas adicionando 10% do fluxo de transações mal-intencionadas e (iii) transações de 20 de dezembro de 2017, onde uma alta aderência à rede foi observado, que denominamos *Ethereum Boom*.

[1]https://etherscan.io/

3.1. Análise dos Resultados do *Penny Attack*

Comparamos o tempo de espera do Ethereum nos três cenários. A Figura 2(b) mostra a distribuição de probabilidades acumulada desses tempos para cada cenário. A curva do Ethereum Boom externa tempos de espera em geral maiores que o *Penny Attack*. Por exemplo, espera por 120 segundos representa 60% das transações com o *Penny Attack*, ao passo que essa espera representa apenas 35% das transações no Ethereum Boom.

Com a Tabela 2 pode-se observar que o ataque aumentou o tempo médio de espera em 42,16%, enquanto o período de maior uso da rede aumentou esse valor em 131,90%. Outro ponto observado é a média máxima do fluxo normal e a média mínima do *Penny Attack* não se encontram. O teste de Kruskal-Wallis foi usado para medir a distinção das curvas de fluxo normal e de ataque, obtendo valor de 60,70 com p-valor próximo a zero, logo o ataque afetou o tempo de espera das transações aqui simuladas.

Tabela 2. Detalhes do tempo de espera em segundos.

	Média	Intervalo de Confiança (95%)	Desvio Padrão	Moda
Normal	79.56	75.15 - 83.96	100.37	19.55
Penny attack	113.11	107.96 - 118.27	126.06	53.54
Ethereum boom	184.50	180.69 - 188.32	122.98	190.09

3.2. Análise dos Modelos de Aprendizado de Máquina

A Tabela 3 mostra a aplicação de diferentes técnicas com subamostragem, sobre-amostragem, e a técnica de SMOTE.

Apresentamos as métricas com a subamostragem, sobre-amostragem e SMOTE respectivamente, onde pode-se observar a importância do estabelecimento do balanceamento de dados, uma vez que as métricas tiveram melhoras significativas, ao exemplo do classificador SVM que teve sua métrica de *recall* adicionada no valor absoluto em no mínimo 0.90. Analisando a métrica de AUC-ROC nos diversos cenários balanceados apresentados, o classificador de Árvore de Decisão obteve maiores valores absolutos (Oversampling e SMOTE). No entanto, ao contrastarmos com métricas de acurácia e precisão, o Random Forest teve um desempenho superior em todos os cenários.

O Comitê *soft* e Comitê *hard* tiveram bons resultados, comparados aos demais classificadores. Apesar disso, não apresentaram as melhores métricas em nenhuma perspectiva de balanceamento.

Observa-se que é possível utilizar de técnicas de aprendizado de máquina para estabelecer modelos classificadores com boas métricas para identificar o *Penny Attack*, ressaltando modelos como a Árvore de Decisão, Random Forest e os Comitês.

4. Conclusão

A partir dos dados obtidos pelas simulações realizadas neste trabalho, podemos ver que o *penny attack* tem o poder de aumentar o tempo de espera de uma transação. Além disso, observamos que a alta demanda por transações por segundo pode gerar uma taxa mais lenta do que o ataque definido aqui.

Além disso, propomos o uso de técnicas de aprendizado de máquina para identificar transações maliciosas, usando não apenas um modelo isolado, mas estabelecendo

Tabela 3. Resultados obtidos pelos métodos de aprendizado de máquina usando diferentes abordagens para balancear os dados.

Balanceamento	Classificador	Acurácia	Precisão	Recall	F_1-score	F_β-score	TP	TN	AUC-ROC
Undersampling	Árvore de Decisão	0.891818	0.454756	0.984925	0.622222	0.798696	196	1766	0.933742
	Random Forest	0.915000	0.515957	0.974874	0.674783	0.827645	194	1819	0.941960
	KNN	0.686364	0.215527	0.934673	0.350282	0.560579	186	1324	0.798171
	SVM Gaussiano	0.614091	0.178854	0.909548	0.298927	0.500553	181	1170	0.747128
	Naive Bayes	0.871364	0.411392	0.979899	0.579495	0.767717	195	1722	0.920235
	Comitê *Soft*	0.907273	0.493606	0.969849	0.654237	0.812974	193	1803	0.935449
	Comitê *Hard*	0.910000	0.501292	0.974874	0.662116	0.819949	194	1808	0.939211
Oversampling	Árvore de Decisão	0.911818	0.506460	0.984925	0.668942	0.828402	196	1810	0.944736
	Random Forest	0.959545	0.757009	0.814070	0.784504	0.801980	162	1949	0.894042
	KNN	0.872727	0.398496	0.798995	0.531773	0.665272	159	1761	0.839527
	SVM Gaussiano	0.685909	0.216590	0.944724	0.352390	0.564904	188	1321	0.802447
	Naive Bayes	0.860909	0.392354	0.979899	0.560345	0.754060	195	1699	0.914487
	Comitê *Soft*	0.915909	0.518919	0.964824	0.674868	0.823328	192	1823	0.937934
	Comitê *Hard*	0.913182	0.510582	0.969849	0.668977	0.821976	193	1816	0.938698
SMOTE	Árvore de Decisão	0.912727	0.509091	0.984925	0.671233	0.829805	196	1812	0.945236
	Random Forest	0.953182	0.703390	0.834171	0.763218	0.804264	166	1931	0.899594
	KNN	0.868182	0.388206	0.793970	0.521452	0.656692	158	1752	0.834766
	SVM Gaussiano	0.690455	0.218458	0.939698	0.354502	0.565981	187	1332	0.802683
	Naive Bayes	0.860909	0.391039	0.964824	0.556522	0.745921	192	1702	0.907699
	Comitê *Soft*	0.913636	0.512064	0.959799	0.667832	0.816938	191	1819	0.934422
	Comitê *Hard*	0.913182	0.510695	0.959799	0.666667	0.816239	191	1818	0.934172

dois comitês, buscando melhores métricas para identificar comportamentos maliciosos no Ethereum. Observa-se que é possível estabelecer modelos classificadores para reconhecer o *Penny Attack* com AUC-ROC maior que 0.945 e F_β maior que 0.82.

Referências

Buterin, V. (2016). Transaction spam attack: Next steps. *URL https://blog.ethereum.org/2016/09/22/transaction-spam-attack-next-steps*.

Chen, T., Li, X., Wang, Y., Chen, J., Li, Z., Luo, X., Au, M. H., and Zhang, X. (2017). An adaptive gas cost mechanism for ethereum to defend against under-priced dos attacks. In *International Conference on Information Security Practice and Experience*, pages 3–24. Springer.

Gautham (2016). Ethereum network comes across yet another dos attack. *URL https://www.newsbtc.com/2016/09/23/ethereum-dao-attack-attack-platforms-credibility*.

Gerard, D. (2017). *Attack of the 50 foot blockchain: Bitcoin, blockchain, Ethereum & smart contracts*. David Gerard.

Sousa, J. E. d. A., Oliveira, V., Valadares, J., Vieira, A. B., Bernardino, H. S., and Dias, G. (2019). An analysis of the fees and pending time correlation in ethereum. In *IFIP LANOMS, 9th Latin American Network Operations and Management Symposium.*

Sousa, J. E. d. A., Oliveira, V., Valadares, J., Vieira, A. B., Villela, S. M., Bernardino, H. S., and Dias, G. (2020). An analysis of the fees and pending time correlation in ethereum. *International Journal of Network Management nem.2113*.