Dizang: Uma solução para coleta de evidências forenses de ataques de injeção na nuvem

Hamilton Fonte II, Marcos A. Simplicio Jr.
Escola Politécnica, Universidade de São Paulo (USP) São Paulo, SP, Brasil
Email: hamiltonii@gmail.com, mjunior@larc.usp.br

***Abstract*—A adoção de arquiteturas em nuvem aumenta a cada dia, e com ela também o número de casos em que esse tipo de tecnologia é usada para fins ilícitos. Infelizmente, devido à natureza volátil da nuvem, a tarefa de coletar evidências para análise forense nesse ambiente tem esbarrado em desafios práticos e legais. Este trabalho analisa propostas na literatura voltadas a resolver os principais desafios existentes na coleta evidências na nuvem, discutindo suas limitações, e então propõe uma solução que cobre coleta, transporte e armazenamento da evidência visando suplantá-las. A solução aqui proposta provê uma forma de correlacionar evidências e sua origem virtual, permitindo transportar e armazenar tais dados sem afetar sua credibilidade. Especificamente, ela tem como focos (1) a reprodutibilidade do processo de coleta e (2) a garantia de custódia da evidência.**

I. Introdução

Técnicas de virtualização, replicação de serviços e compartilhamento de recursos entre múltiplos usuários (multi-inquilinato) proveem a nuvens computacionais uma elevada escalabilidade [1]. Ao mesmo tempo, tais mecanismos também criam uma elevada volatilidade dos recursos virtuais que executam aplicações em nuvem. Afinal, quando submetida a uma carga elevada, uma aplicação hospedada na nuvem pode criar clones das máquinas virtuais (*virtual machines* – VMs) que a hospedam e balancear a carga entre elas, de modo a atender à demanda sem prejuízos na qualidade do serviço oferecido. Após esse pico, as máquinas que foram clonadas são normalmente desativadas, seus recursos liberados e o sistema retorna à capacidade anterior, evitando-se custos desnecessários

Embora interessante do ponto de vista de eficiência e custos, do ponto de vista forense a volatilidade da nuvem traz problemas em caso de ataques. Por exemplo, caso uma das instâncias de processamento virtuais criadas temporariamente seja alvo de ameaças que atuam diretamente na sua memória, sem deixar rastros em discos (e.g., arquivos de *log*), as evidências desse evento podem ser completamente perdidas após elas serem desativadas e terem seus recursos liberados. Essa dificuldade é ainda agravada por aspectos como multi-inquilinato e multi-jurisdição típicas de soluções em nuvem [2]. Especificamente, o aspecto multi-inquilino dificulta a obtenção do *hardware* que executa as aplicações de interesse, pois, como ele é compartilhado por vários usuários, removê-los para análise poderia levar a uma violação de privacidade dos usuários não relacionados à investigação. Já a natureza distribuída da nuvem pode levar à alocação de informações relevantes à investigação em vários países, dificultando a obtenção das mesmas em especial quando não existem acordos de cooperação entre as entidades envolvidas [3]. Combinadas, tais características dificultam a coleta de evidências com a credibilidade necessária para que elas possam ser usadas em processos legais, o que exige o respeito à privacidade, à jurisdição e à cadeia de custódia, bem como a reprodutibilidade do processo de coleta [4].

Embora existam soluções na literatura que abordam a coleta de informações de nuvem com o propósito de análise forense, a grande maioria delas aborda a coleta, o transporte e o armazenamento de forma isolada. Por exemplo, trabalhos como [5] e [6] tratam de fatores como multi-inquilinato e multi-jurisdição, discutindo formas de coleta e preservação da evidência fora da nuvem. Já estudos como [7] se concentram na análise forense para a coleta de evidência de máquinas virtuais enquanto elas estão em execução, enquanto trabalhos como [8] abordam a questão de processos de garantia de cadeia de custódia em ambientes de nuvem para transporte da evidência. Por outro lado, não foram identificadas na literatura propostas que (1) descrevam como o dado é coletado e armazenado observando a cadeia de custódia, e (2) visem garantir que, mesmo que um recurso virtualizado (e.g., uma VM) seja desalocada, haja condições de se reproduzir o processo de coleta de evidências.

O presente trabalho visa suplantar tais limitações por meio de uma proposta que tem como focos (1) a reprodutibilidade do processo de coleta, (2) o estabelecimento de vínculo entre a evidência coletada e sua origem, (3) a preservação da jurisdição e da privacidade dos não envolvidos na investigação e (4) a garantia de custódia da evidência. Em suma, a solução descrita provê uma forma de correlacionar evidências e sua origem virtual, permitindo transportar e armazenar tais dados de modo a preservar sua credibilidade. Para isso, a proposta supõe que o sistema sendo monitorado é executado dentro de um contêiner em nuvem. O foco da solução em contêiner se justifica pelo crescimento da adoção de conteinerização nos últimos anos, bem como pela previsão de que este será o modelo de implementação mais usado em aplicações futuras [9]. A solução tem como alvo específico ataques de injeção de código [10], pois estes, quando usados contra uma arquitetura em nuvem, não deixam rastros quando recursos de processamento virtuais são desativados e sua memória é liberada [11], [10]. Em particular, têm especial interesse quatro tipos específicos dessa família de ameaças [10]:

- **Injeção remota de bibliotecas**: Um processo malicioso força o processo alvo a carregar uma biblioteca em seu espaço de memória. Como resultado, o código da biblioteca carregada executa com os mesmos privilégios do executável em que ela foi injetada. Esta estratégia, comumente usada para instalar malwares, pode fazer com que uma biblioteca maliciosa armazenada no sistema seja distribuída por vários processos de uma mesma máquina, dificultando sua remoção [12].
- **Inline Hooking**: Um processo malicioso escreve código como uma sequência de bytes diretamente no espaço de memória de um processo alvo, e então força este último a executar o código injetado. O código pode ser, por exemplo, um *script* de *shell*.
- **Injeção reflexiva de biblioteca**: Um processo malicioso acessa diretamente a memória do processo alvo, inserindo nela o código de uma biblioteca na forma de uma sequência de bytes, e então força o processo a executar essa biblioteca. Nessa forma de ataque, a biblioteca maliciosa não existe fisicamente; isso torna tal estratégia de injeção de código potencialmente mais atrativa, pois o carregamento da biblioteca não é registrado no sistema operacional (SO), dificultando a detecção do ataque [13].
- **Injeção de processo vazio**: Um processo malicioso dispara uma instância de um processo legítimo no estado "suspenso"; a área do executável é então liberada e realocada com código malicioso.

O restante deste documento está organizado da seguinte forma. A Seção II discute brevemente soluções em nuvem e suas características. A Seção III analisa os trabalhos relacionados na área de forense de memória. A Seção IV detalha a solução proposta e avalia como ela trata os desafios alvo deste projeto. Finalmente, a Seção V apresenta algumas considerações finais e discute ideias para trabalhos futuros.

II. Adoção de arquiteturas em nuvem e contêineres

Uma nuvem computacional é um modelo de infraestrutura no qual recursos compartilhados em quantidade configurável, acessíveis via rede, são alocados e desalocados com esforço mínimo de gerenciamento por parte de um provedor de serviços. [14] Há três modelos principais de comercialização de uso da nuvem [14]: *software* como serviço (*Software as a Service* – SaaS), na qual se provê o *software* que será usado pelo cliente; plataforma como serviço (*Platform as a Service* – PaaS), na qual se provê o ambiente para que o cliente desenvolva, teste e execute seu *software*; e, o tipo mais pertinente para este trabalho, Infraestrutura como serviço (*Infrastructure as a Service* – IaaS), na qual são fornecidos recursos computacionais básicos, como processamento e memória, em geral de forma virtualizada.

A virtualização de recursos na nuvem, embora tradicionalmente feita por meio de máquinas virtuais, vêm sendo crescentemente feita também na forma de contêineres. De fato, segundo o "Container Market Adoption Survey 2016", realizado pelas empresas DevOps.com (https://devops.com/) e ClusterHQ (https://clusterhq.com) com 235 empresas que têm desenvolvimento de software como sua atividade fim ou como suporte à atividade fim, 76% dos respondentes utilizam contêiners para melhorar a eficiência do processo de desenvolvimento e em suas arquiteturas de micro-serviços em nuvem. Diferentemente de máquinas virtuais, que envolvem a criação de um *hardware* virtual e de um sistema operacional (SO) acima do sistema nativo e que opera independente deste, a virtualização com contêineres é feita no nível do SO nativo, tem uma implementação mais simples eliminado camadas entre o aplicativo executado e o *hardware* físico. Uma tecnologia bastante utilizada para esse propósito são Contêiners Linux (LXC) [15], que aproveitam-se de funcionalidades como cgroups e namespacing do kernel do Linux para auxiliar no gerenciamento e isolamento de recursos virtuais.

III. Trabalhos relacionados

Existem vários aspectos relativos à análise forense na nuvem, indo desde a coleta de informações até a garantia da cadeia de custódia de evidências. Para uma discussão mais estruturada dos trabalhos disponíveis na literatura sobre o tema, a seguir eles são apresentados com base nos diferentes aspectos que abordam.

A. Acessar e coletar as informações de memória das máquinas virtuais em nuvem

Diversos trabalhos de análise forense na nuvem se concentram na coleta de dados "após o fato", ou seja, após a intrusão ser detectada [6], [16], [5], [7], [8]. Os processos de coleta descritos nesses trabalhos podem ser iniciados de forma manual ou automaticamente, via integração com um mecanismo de detecção de intrusão. No caso específico de memória volátil, tal forma de coleta não consegue descrever como era a memória antes da intrusão, pois o processo só é acionado depois da detecção do ataque. Tal limitação pode trazer prejuízos à investigação, dado que algumas análises dependem exatamente da capacidade de se comparar dois momentos da memória [10]. Entre os trabalhos estudados, a única proposta encontrada que leva tal necessidade em consideração é [17], que propõe que o dado seja armazenado no próprio equipamento sob análise. Infelizmente, entretanto, a aplicação de tal abordagem no cenário em nuvem é pouco viável, pois pode levar à perda de informações importantes caso a máquina virtual ou contêiner seja desativada, tendo seus recursos liberados.

Existem ainda trabalhos voltados à coleta de informações durante a execução do sistema, nos quais os dados são constantemente coletados sem distinção do que aconteceu antes ou depois do fato de interesse. Esse é o caso de trabalhos como [16], [5], [8], que adotam a estratégia de isolar e parar a máquina virtual para em seguida realizar o processo de coleta. Embora interessantes, as abordagens descritas nesses trabalhos podem levar a um elevado volume de dados coletados, além de também não tratarem o cenário em que é necessário coletar evidências quando os recursos virtuais contendo tais informações são liberados.

B. Capacidade de reproduzir o processo e obter os mesmos resultados

Se, durante uma análise forense, analistas diferentes obtêm resultados distintos ao executar o mesmo procedimento de coleta, a evidência gerada não tem credibilidade, inviabilizando seu uso em um processo legal. Por essa razão, a reprodutibilidade do processo de coleta é uma parte importante da geração de evidências para análise forense. Infelizmente, entretanto, nenhuma das propostas encontradas na literatura atualmente permite tal reprodutibilidade em cenários de nuvem em que máquinas virtuais ou contêineres são desativados e seus recursos físicos liberados: todas elas dependem da existência do recurso virtual para a repetição do processo de coleta.

C. Não violar privacidade ou jurisdição das partes não envolvidas na investigação

Em um ambiente de nuvem pública, remover o *hardware* para análise posterior pode levar à violação de privacidade de usuários, uma vez que o multi-inquilinato desse cenário faz com que uma mesma máquina física guarde informações de diversos clientes, alguns dos quais podem não estar envolvidos na investigação em curso. Diversos trabalhos na literatura tratam esse problema adequadamente, por meio das duas estratégias principais: a primeira, adotada em [6], [7], [16], [5], consiste em coletar dados pertinentes à investigação e armazená-los fora da nuvem; a segunda, empregada em [8] e que constitui um caso específico de [7], depende da cooperação do provedor de serviços de nuvem para conseguir as informações necessárias à investigação. Depender do provedor de serviços de nuvem é uma estratégia pouco recomendada, entretanto, pois (1) o volume de dados de usuários pode forçar os provedores a limitar o tamanho dos *logs* armazenados, e (2) caso ocorra uma indisponibilidade causada por um ataque, o objetivo do provedor será o de restabelecer o serviço, não necessariamente o de preservar evidências[18].

D. Garantir a cadeia de custódia da evidência

Dentre os trabalhos analisados, apenas [8] aborda a questão da garantia da cadeia de custódia. Especificamente, o trabalho emprega *hashes* para verificar a integridade da evidência, permitindo a detecção de alterações na mesma, embora não explique os mecanismos que poderiam ser utilizados para impedir acesso não autorizado (e, assim, potencial alteração) aos próprios hashes. As propostas dos outros autores concentram-se apenas no aspecto técnico da coleta, sem discutir claramente garantia de custódia mas apenas mencionando que as evidências devem ser coletadas de forma "forensicamente aceitável".

E. Resumo

A Tabela I mostra um comparativo das soluções estudadas, considerando os aspectos discutidos nesta seção, posicionando as contribuições da proposta apresentada neste trabalho.

TABLE I
COMPARATIVO DE SOLUÇÕES DE COLETA DE INFORMAÇÕES DE MEMÓRIA DE MÁQUINAS EM NUVEM PARA ANÁLISE FORENSE

	Coleta é contínua?	**Reproduz o processo sem a VM?**	**Garante cadeia de custódia?**	**Preserva jurisdição e privacidade?**
Dizang(esta proposta)	✓	✓	✓	✓
[7]	✘	✘	✘	✓
[16]	✘	✘	✘	✓
[5]	✘	✘	✘	✓
[19]	✘	✘	✘	✓
[6]	✘	✘	✓	✓
[8]	✓	✘	✓	✓
[20]	✘	✘	✘	✓
[21]	✘	✘	✘	✓
[17]	✓	✘	✘	✓
[22]	✓	✘	✓	✓

IV. SOLUÇÃO PROPOSTA: DIZANG

A presente proposta tem como objetivo principal coletar memória de recursos computacionais virtuais de modo a conseguir: (1) identificar a fonte da evidência, mesmo se o recurso virtual não existir mais; (2) descrever o sistema antes e depois do incidente; (3) transportar e armazenar a memória coletada de uma forma que garanta sua integridade e confidencialidade; e (4) não violar a jurisdição e a privacidade de outros usuários que porventura tenham recursos alocados no mesmo servidor físico. A solução aqui apresentada, denominada Dizang, é descrita em detalhes a seguir.

A. Descrição

Em sistemas computacionais executados sobre uma infraestrutura física (i.e., não virtualizada), pode-se fazer uma associação direta entre um recurso qualquer, como uma informação da memória, imagem de disco ou pacotes trafegando na rede, e sua origem correspondente. Já em sistemas construídos sobre uma infraestrutura virtual, em especial quando esta é auto-escalável, os recursos computacionais são altamente voláteis e, portanto, podem ser desalocados a qualquer momento. Para conseguir correlacionar uma evidência a

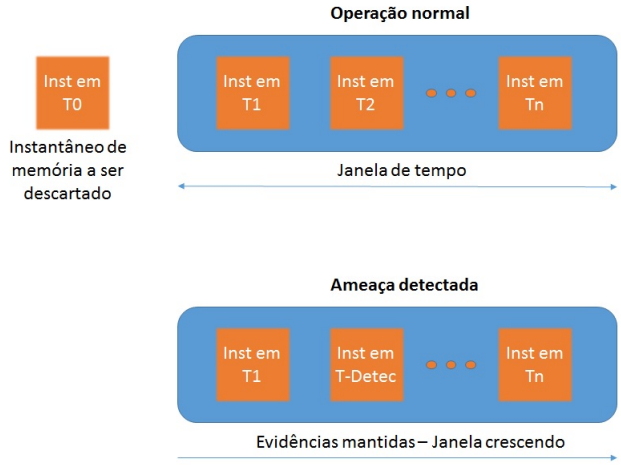
sua origem volátil, é necessário utilizar outro elemento em que persista a relação fonte-evidência, o que na presente proposta é feito por meio de contêineres linux (LXC). Embora um contêiner seja um *software* e, portanto, também volátil, cada imagem compilada e sua execução na forma de contêiner são normalmente atrelados a um *hash* que identifica univocamente essa relação. O contêiner também permite identificar com mais precisão a fonte de uma evidência, uma vez que é possível dividir as partes de um sistema em contêineres: por exemplo, um contêiner para o motor de páginas dinâmicas (e.g., Apache [23]), outro com a lógica de negócios (e.g., *golang* [24]) e um terceiro para um banco de dados (e.g., *Cassandra* [25]).

A cópia de memória não é uma atividade atômica, pois ela é executada em conjunto com outros processos. Portanto, caso um desses processos seja um código malicioso apagando traços de sua existência da memória do contêiner, informações possivelmente importantes para a investigação podem acabar sendo perdidas. Com o objetivo de deixar o processo de cópia da memória mais atômico, Dizanginterrompe temporariamente a execução do contêiner, realiza a cópia de sua memória, e em seguida retoma sua execução. Essa técnica, que é semelhante àquela adotada em [26] para máquinas virtuais, produz um instantâneo da memória volátil do contêiner; isso permite sua análise em um estado de repouso, ou seja, sem a necessidade de ter o contêiner em execução. Ao realizar a coleta em intervalos de tempo adequados, é possível construir um histórico do estado da memória durante a execução no contêiner.

A maioria das técnicas forenses mais usadas atualmente são voltadas à obtenção da informação em sua totalidade, seja via cópia bit a bit, seja por meio da obtenção do *hardware* físico [27] [28]. Embora tais técnicas possam parecer interessantes à primeira vista, elas muitas vezes acabam sendo responsáveis por um problema: o crescente volume de informações que os investigadores precisam analisar [29]. Para mitigar essa dificuldade, em Dizangsão adotadas duas estratégias: a primeira é a definição de um volume de dados que possa ser considerado *suficiente* para a realização de uma investigação; a segunda é a definição de uma *idade máxima* para a evidência enquanto o sistema trabalha em condições normais, isto é, quando não está sob ataque. Para detectar e analisar intrusões na memória de processos, é necessário ter uma cópia da memória antes e depois da intrusão [10]. Assim, a solução proposta implementa uma janela de instantâneos de memória cobrindo um intervalo de tempo pré-definido, como ilustrado na Figura 1. Em condições normais de operação, as evidências são coletadas com certa periodicidade e coletas que atingem uma determinada idade são descartadas. Em contraste, após a detecção de um evento de ataque (e.g., por um sistema de detecção de intrusões), Dizangdeixa de descartar as coletas mais antigas do *log* de monitoramento, sendo possível conhecer o sistema antes e depois do ataque e, assim, avaliar sua evolução.

Para persistir a relação evidência-origem e garantir a integridade da mesma, a presente proposta calcula o hash do par [evidência, identificador da imagem do contêiner] e

Fig. 1. Janela deslizante de coleta de evidência



armazena a tripla [hash, identificador da imagem do contêiner, evidência]. Para evitar eventuais problemas com o armazenamento desses dados em países com jurisdições diferentes daquelas que devem ser aplicadas na investigação em questão, as evidências coletadas são armazenadas em um local físico fora da nuvem, após serem transportadas por meio de um canal seguro (e.g., usando o protocolo *Transport Layer Security* – TLS [30]).

B. Implementação

Os métodos propostos foram implementados em uma plataforma de testes visando avaliar a eficácia de Dizangem coletar as informações de memória dos contêineres de forma reprodutível, garantindo a cadeia de custódia sem violar jurisdições ou a privacidade de usuários. A solução, ilustrada na Figura 5, consistiu na criação de 1 máquina virtual usando o Oracle Virtual Box 5.0 [31] em um notebook Intel i5 de 2.30Mhz e 4Gb de RAM com sistema operacional de 64 bits. Essas máquinas virtuais possuem 2 Gb de memória RAM e emulam apenas 1 processador, e em cada uma delas foi instalado o Docker Engine 1.10 [32] e a API Docker 1.21 [32], com os quais foram criados 3 contêiners executando o nginx 1.0 [33] em diferentes portas. Usando uma aplicação Java que descobre qual o identificador de processo associado a cada contêiner, pode-se copiar conteúdo do *descritor de alocação de memória não uniforme* (**/proc/pid/numa_maps**) que contem a alocação das páginas de memória, os nós que estão associados a essas páginas, o que está alocado e suas respectivas políticas de acesso[34]. TODO: explique o que são esses dados! Hamilton - Feito. A cópia e gravação desse arquivo acontece da seguinte forma: a cada minuto, a aplicação (1) pausa o contêiner em questão, (2) copia a diretório **numa_maps**, (3) concatena os dados obtidos com o *hash* de identificação da imagem do contêiner, (4) calcula o *hash* do conjunto e (5) salva o resultado em um arquivo cujo nome é o identificador da imagem do contêiner e a extensão é **.mem**. Após a conclusão do processo de cópia, a mesma aplicação verifica se existem arquivos **.mem** em disco mais

antigos do que um certo intervalo de tempo “t”, descartando-os.

TODO: IMPORTANTE: discuta em alguns parágrafos os resultados. Até aqui, você só disse o que fez, mas não falou nada sobre o que isso traz de impactos positivos ou negativos. Por exemplo: quanto tempo demora a cópia (isso pode ser crítico para um processo que opera em tempo real, já que ele será pausado...)? Enfim, tire conclusões dos experimentos, porque senão os experimentos não têm qualquer utilidade para o leitor... Quanto mais completa a discussão (não só “tempos”, mas também memória, quantidade de dados gerados por hora, uso de banda, etc.), melhor. Gráficos e tabelas com dados obtidos são muito bem vindos. - Hamilton: Feito

TODO: IMPORTANTE: Discuta também a relação entre o que foi feito e os ataques de injeção mencionados na introdução. O seu texto inicialmente menciona tais ataques como parte do seu foco, mas nunca mais retoma a existência deles depois que você descreve a solução... Não precisa ser uma discussão longa, mas ao menos algumas considerações sobre o porquê de você conseguir analisar tais ataques usando a técnica proposta - Hamilton: Feito.

C. Experimento e Conclusões

Para verificar se a presente proposta atinge os aos objetivos declarados montou-se o seguinte experimento. A solução Dizangfoi configurada para realizar coletas de memória em intervalos de 1 minuto e apagar aquelas que tivessem mais de 5 minutos de existência. Permitiu-se que ela rodasse por 30 minutos. Durante este tempo foram coletadas métricas de uso de espaço em disco utilizado pelos instantâneos de memória salvos e o tempo de pausa no contêiner usado para a cópia das mesmas. A cada coleta executou-se o comando `du -sh *.mem` do *Unix* para retornar a lista dos instantâneos e o espaço em disco ocupado pelas coletas. Ao fim do experimento removeu-se os contêiners.

O espaço em disco ocupado pelos instantâneos de memória durante o experimento é mostrado na tabela II e plotado no gráfico 2. O gráfico mostra que o aumento do uso do espaço em disco é linear e o crescimento se interrompe quando é atingido o limite de tempo configurado para a janela pois as coletas com tempo de vida maior que tal limite são apagadas do disco. Podemos concluir que a solução mantem sob controle o espaço em disco ocupado pelas coletas.

Fig. 2. Evolução do uso do espaço em disco

TABLE II
EVOLUÇÃO DO USO DO ESPAÇO EM DISCO

Tamanho total ocupado (KBytes)	Tempo (segundos)
240	1
480	2
720	3
960	4
1200	5
1200	6
1200	7
1200	8
1200	9
1200	10
1200	11
1200	12
1200	13
1200	14
1200	15
1200	16
1200	17
1200	18
1200	19
1200	20
1200	21
1200	22
1200	23
1200	24
1200	25
1200	26
1200	27
1200	28
1200	29
1200	30

A figura 3 é uma listagem de alguns dos instantâneos de memória salvos pela solução. Nela podemos ver que as coletas continuaram no disco da máquina mesmo após a remoção dos contêiners. Usando o identificador do contêiner e da imagem como nome do arquivo que armazena o instantâneo da memória conseguimos associar a evidência a sua origem.

Fig. 3. Lista de instantâneos de memória

No evento da detecção de uma ameaça a presente proposta deixa de apagar as coletas mais antigas. Desta forma é capaz de descrever a história das alterações da memória do contêiner e com isso viabilizar a análise forense em busca das 4 vulnerabilidades de injeção de código citadas no início do artigo pois consegue descrever o estado do sistema antes e depois do incidente [10].

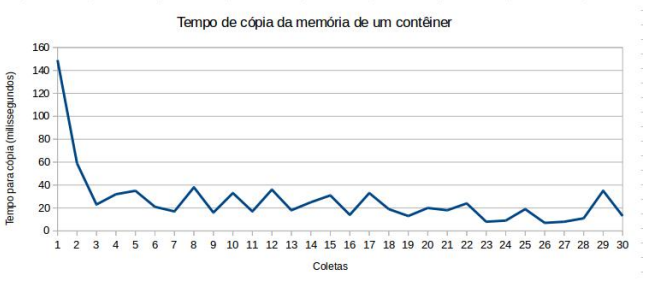
Uma preocupação da presente proposta é o impacto que a pausa de um contêiner para coleta de dados pode causar na performance da aplicação executando no mesmo. A tabela III mostra os tempos transcorridos para a cópia de memória durante o experimento. Os dados foram plotados no gráfico 4. É possível notar que após a inicialização, o tempo para realizar a

cópia da memória pela presente proposta no ambiente descrito na seção *implementação* varia entre 20 e 40 milisegundos. Para avaliar o tempo de cópia deve-se levar em consideração o que está sendo executado no contêiner. Na implementação do presente experimento, cada contêiner está executando um motor de páginas web dinâmicas nginx 1.0 [33]. Neste cenário o tempo gasto com a cópia é adequado.

TABLE III
TEMPO DE CÓPIA DA MEMÓRIA DE UM CONTÊINER

Tempo de cópia (Milisegundos)	Coleta
149	1
59	2
23	3
32	4
35	5
21	6
17	7
38	8
16	9
33	10
17	11
36	12
18	13
25	14
31	15
14	16
33	17
19	18
13	19
20	20
18	21
24	22
8	23
9	24
19	25
7	26
8	27
11	28
35	29
13	30

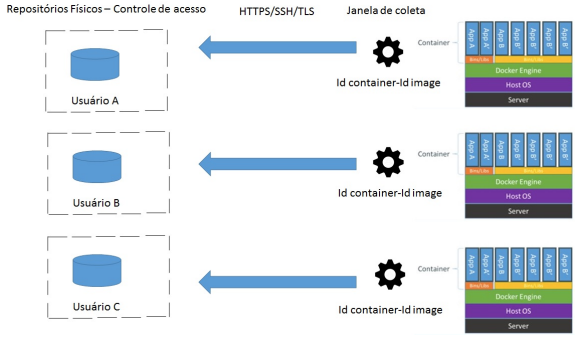
Fig. 4. Lista de instantâneos de memória



D. Limitações

Como a solução descrita tem como foco coletar informações de memória no espaço do usuário (*user space*), ela não consegue acessar o espaço de kernel (*kernel space*). Assim, Dizangem princípio não provê suporte a técnicas de investigação de malware que se baseiam em informações do *kernel space*, como, por exemplo, a comparação de informações do bloco do ambiente do processo (*Process Environment Block – PEB*), que ficam no *user space*, com informações do descritor de endereços de memória virtual (*Virtual Address Descriptor – VAD*), que fica no *kernel space*. Análise de ameaças que realizam manipulação direta dos objetos do kernel (D.K.O.M.– *Direct Kernel Object Manipulation*) também não se beneficiam com a solução aqui proposta. TODO: Não entendi a "associação com o contêiner" aqui. Você quer dizer que "não se beneficiam com a solução aqui proposta" ou outra coisa? Você não definiu o que seria a tal "associação com o contêiner" fora do contexto da sua solução, então ficou confuso - Hamilton: Feito.

Fig. 5. Desenho completo da solução Dizang TODO: "Solução completa" do que, cara pálida? Legendas devem ser auto-explicativas (i.e., se o leitor não ler sequer uma linha do seu texto e bater o olho em uma figura, ele deve conseguir entender alguma coisa daquela figura...) Com o nome "Solução completa", pode ser desde uma "receita de bolo" até a "cura para o câncer"... - Hamilton: Feito



V. CONSIDERAÇÕES FINAIS

TODO: Uma boa conclusão retoma, logo na primeira frase, o problema que ela se propunha a resolver. Comece com uma ou mais frases nesse sentido, (nota: **sem** copiar+colar de outro ponto do texto). - Hamilton: Feito A proposta apresentada é capaz de relacionar o instantâneo de memória a sua origem utilizando o *hash* calculado da imagem do contêiner como identificador da evidência armazenada, limita a quantidade de dados armazenados usando a implementação de janela de armazenamento e é capaz de descrever a memória antes e depois de um ataque viabilizando a análise de ataques de injeção de memória. Combinada com uma ferramenta adequada para análise das ameaças, essa característica de Dizangpode ser uma ferramenta poderosa para análises forenses. Cabe notar, entretanto, que muitas ferramentas de análise malware disponíveis no mercado necessitam que todo o conteúdo da memória da máquina esteja disponível para realização da análise com o FROST [5], Volatility Framework [35] e FATkit [36] (e.g., TODO: Dê exemplos com referências! - Hamilton: Feito), o que inviabiliza seu uso diretamente sobre a memória de processos conforme coletado por Dizang. Portanto, como trabalho futuro, pretende-se desenvolver uma ferramenta mais flexível de análise, que, embora potencialmente utilize técnicas similares às atuais para detecção de ameaças, seja capaz de fazê-lo sem necessariamente ter acesso à memória completa da máquina.

COMMENT: As próximas frases, como escritas, são "tiros no pé": você fica falando das limitações que não são da sua proposta, mas sim do universo onde sua proposta se encaixa... Deixe a propaganda negativa para seus concorrentes fazerem, não para você... Refraseei para manter o mesmo conteúdo, mas de forma positiva em vez de negativa.

REFERENCES

[1] A. M. Morsy, J. Grundy, and I. Muller, "An Analysis of the Cloud Computing Security Problem," in *APSEC Cloud Workshop*, Sydney, Australia, 2010. [Online]. Available: https://arxiv.org/abs/1609.01107

[2] R. Keyun, C. Joe, K. Tahar, and C. Mark, *Advances in Digital Forensics IV*, 7th ed., P. Gilbert and S. Sujeet, Eds., Orlando, 2011, vol. 1.

[3] J. Dykstra and A. T. Sherman, "Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques," *Digital Investigation*, vol. 9, pp. S90–S98, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2012.05.001

[4] S. Rahman and M. N. A. Khan, "Review of live forensic analysis techniques," *International Journal of Hybrid Information Technology*, vol. 8, no. 2, pp. 379–388, 2015. [Online]. Available: http://www.sersc.org/journals/IJHIT/

[5] J. Dykstra and A. T. Sherman, "Design and implementation of FROST: Digital forensic tools for the OpenStack cloud computing platform," *Digital Investigation*, vol. 10, pp. S87–S95, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2013.06.010

[6] Z. Reichert, K. Richards, and K. Yoshigoe, "Automated forensic data acquisition in the cloud," *Proceedings - 11th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2014*, pp. 725–730, 2015.

[7] S. George, H. Venter, and F. Thomas, "Digital Forensic Framework for a Cloud Environment," in *IST Africa 2012*, P. Cunningham and M. Cunningham, Eds. Tanzania: Internation Information Management Corporation, 2012, pp. 1–8.

[8] T. Sang, "A log-based approach to make digital forensics easier on cloud computing," *Proceedings of the 2013 3rd International Conference on Intelligent System Design and Engineering Applications, ISDEA 2013*, pp. 91–94, 2013.

[9] S. F. Piraghaj, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "A Framework and Algorithm for Energy Efficient Container Consolidation in Cloud Data Centers," *Proceedings - 2015 IEEE International Conference on Data Science and Data Intensive Systems; 8th IEEE International Conference Cyber, Physical and Social Computing; 11th IEEE International Conference on Green Computing and Communications and 8th IEEE International Conference on Internet of Things, DSDIS/CPSCom/GreenCom/iThings 2015*, pp. 368–375, 2016.

[10] A. Case, M. Ligh, L. Jamie, and A. Walters, *The Art of Memory Forensics: Detecting malware and threats in Windows, Linux and Mac memory*, kindle ed. Wiley, 2014.

[11] S. Vömel and J. Stüttgen, "An evaluation platform for forensic memory acquisition software," in *The Digital Forensic Research Conference*, vol. 10, Monterey, CA, 2013, pp. S30–S40.

[12] M. Miller and J. Turkulainen, "Remote Library Injection," pp. 1–40, 2004.

[13] B. S. Fewer, "Reflective DLL Injection," no. October, 2008.

[14] P. Mell and T. Grance, "The NIST definition of cloud computing," *NIST Special Publication*, vol. 145, p. 7, 2011. [Online]. Available: http://www.mendeley.com/research/the-nist-definition-about-cloud-computing/

[15] Linuxcontainers.org, "Linux Containers (LXC)," 2015. [Online]. Available: https://linuxcontainers.org/lxc/introduction/

[16] R. Poisel, E. Malzer, and S. Tjoa, "Evidence and cloud computing: The virtual machine introspection approach," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 4, no. 1, pp. 135–152, 2013. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.469.937

[17] F. N. Dezfouli, A. Dehghantanha, R. Mahmoud, N. F. Binti Mohd Sani, and S. Bin Shamsuddin, "Volatile memory acquisition using backup for forensic investigation," *Proceedings 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic, CyberSec 2012*, pp. 186–189, 2012.

[18] N. L. Clarke, C. Reich, A. Saad, and S. Furnell, "Cloud Forensics : A Review of Challenges , Solutions and Open Problems Cloud Forensics : A Review of Challenges , Solutions and Open Problems," no. April, pp. 1–9, 2015.

[19] D. Barbara, "Desafios da perícia forense em um ambiente de computação nas nuvens," Universidade do Planalto Catarinense, Tech. Rep., 2014.

[20] B. Dolan-Gavitt, T. Leek, M. Zhivich, J. Giffin, and W. Lee, "Virtuoso: Narrowing the semantic gap in virtual machine introspection," *Proceedings - IEEE Symposium on Security and Privacy*, pp. 297–312, 2011.

[21] A. Aljaedi, D. Lindskog, P. Zavarsky, R. Ruhl, and F. Almari, "Comparative Analysis of Volatile Memory Forensics," *IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT) and IEEE International Conference on Social Computing (SocialCom)*, pp. 1253–1258, 2011.

[22] R. B. van Baar, H. M. A. van Beek, and E. J. van Eijk, "Digital Forensics as a Service: A game changer," *Digital Investigation*, vol. 11, pp. S54–S62, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2014.03.007

[23] Apache, "Apache-Tomcat," 2016. [Online]. Available: http://tomcat.apache.org/

[24] Google, "Golang." [Online]. Available: https://golang.org/

[25] Apache, "Apache-Cassandra," 2016. [Online]. Available: http://cassandra.apache.org/

[26] M. Rafique and M. N. A. Khan, "Exploring Static and Live Digital Forensics: Methods, Practices and Tools," *International Journal of Scientific & Engineering Research*, vol. 4, no. 10, pp. 1048–1056, 2013. [Online]. Available: http://www.ijser.org/researchpaper%5CExploring-Static-and-Live-Digital-Forensic-Methods-Practices-and-Tools.pdf

[27] S. Simou, C. Kalloniatis, E. Kavakli, and S. Gritzalis, "Cloud forensics: Identifying the major issues and challenges," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8484 LNCS, pp. 271–284, 2014.

[28] D. Bem, F. Feld, E. Huebner, and O. Bem, "Computer Forensics - Past , Present and Future," *Journal of Information Science and Technology*, vol. 5, no. 3, pp. 43–59, 2008.

[29] D. Quick and K. K. R. Choo, "Impacts of increasing volume of digital forensic data: A survey and future research challenges," *Digital Investigation*, vol. 11, no. 4, pp. 273–294, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2014.09.002

[30] R. E. Dierks T, "The Transport Layer Security (TLS) Protocol," pp. 1–104, 2008.

[31] Oracle, "Virtual Box 5.1." [Online]. Available: https://www.virtualbox.org/

[32] Docker Inc, "Docker." [Online]. Available: https://www.docker.com/

[33] S. Igor, "NginX." [Online]. Available: https://nginx.org/

[34] Unix Man Pages, "Numa Maps - Non Uniform Memory Architecture." [Online]. Available: http://man7.org/linux/man-pages/man7/numa.7.html

[35] Volatility Foundation, "Volatility Framework," 2014. [Online]. Available: http://www.volatilityfoundation.org/

[36] T. Fraser, W. A. Arbaugh, N. L. Petroni Jr, and W. Aaron, "FATKit: A framework for the extraction and analysis of digital forensic data from volatile system memory," *Digital Investigations*, vol. 3, pp. 197–210, 2006.