HAMILTON FONTE II

DIZANG: UMA SOLUÇÃO PARA COLETA DE EVIDÊNCIAS FORENSES PARA ATAQUES DE INJEÇÃO NA NUVEM

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do Título de Mestre em Engenharia de Computação.

São Paulo
2019

HAMILTON FONTE II

DIZANG: UMA SOLUÇÃO PARA COLETA DE EVIDÊNCIAS FORENSES PARA ATAQUES DE INJEÇÃO NA NUVEM

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do Título de Mestre em Engenharia de Computação.

Área de concentração:
Engenharia da Computação

Orientador:
Marcos Antonio Simplicio Junior

São Paulo
2019

A minha esposa e filha, pelas noites e finais de semana que não estive com vocês durante a elaboração deste trabalho

AGRADECIMENTOS

As Deus por ter dado sua benção durante todo momento para que eu pudesse continuar meus estudos.

A minha familia pelo apoio incondicional

Ao meu orientador o Prof. Dr. Marcos Simplício, que me amparou e conduziu nesta jornada.

A Universidade de São Paulo, a Escola Politécnica e o Programa de Pós Graduação, por ter me acolhido como aluno de mestrado.

RESUMO

A adoção de arquiteturas em nuvem aumenta a cada dia, e proporcionalmente aumenta 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. Mais precisamente, a prática herdada da forense tradicional para coleta de evidências, por meio da qual se isola a cena do crime e coletam-se todas as evidências, foi traduzida para a forense digital como a cópia bit a bit da mídia que se deseja investigar. Tal prática leva à coleta de grandes volumes de informação para análise, impactando negativamente o tempo de investigação. Além disso, duas características das soluções em nuvem dificultam a obtenção de evidências forensicamente confiáveis, além de atrapalhar o processo de análise de incidentes de modo geral. A primeira é que o compartilhamento de recursos físicos entre vários usuários impede a sua remoção para análise, uma vez que isso violaria a privacidade de indivíduos não envolvidos na investigação. A segunda é que a localização do recurso físico em uma região geográfica diferente daquela onde o crime foi cometido pode impedir a coleta de evidências caso não haja acordos de cooperação estabelecidos. Estes aspectos, se não forem levados em consideração, podem colocar em xeque a credibilidade das evidências e diminuir a chance de serem aceitas em um processo legal. Este trabalho analisa propostas na literatura voltadas a resolver tais desafios na coleta evidências na nuvem, discutindo suas limitações. Propõe-se então uma solução que cobre coleta, transporte e armazenamento da evidência, visando suplantar as limitações existentes no estado da arte. A solução proposta, denominada Dizang , provê uma forma de correlacionar evidências e sua origem virtual, permitindo transportar e armazenar tais dados sem afetar sua credibilidade. Para tal, é proposto um mecanismo de identificação única do gerador da evidência, permitindo então preservar a relação evidência-origem mesmo que esta última não exista mais no ambiente sob investigação. Em resumo, Dizang tem como focos (1) a reprodutibilidade do processo de coleta, (2) o estabelecimento de um vínculo entre a evidência coletada e sua origem, (3) a preservação da jurisdição e da privacidade de usuários não envolvidos na investigação e (4) a garantia de custódia da evidência.

ABSTRACT

The adoption of cloud architectures is increasing day by day, and proportionally increases the number of cases in which this technology is used for illicit purposes. Unfortunately, due to the volatile nature of the cloud, the task of collecting evidences for forensic analysis in this environment has run into practical and legal challenges. The practice inherited from the traditional forensics, through which the crime scene is isolated and all the evidence is collected, was carried to digital forensics as the bitwise copy of the media to be investigated. Such practice leads to the collection of large volumes of information for analysis, negatively impacting the investigation time. In addition, two characteristics of cloud solutions make it difficult to obtain forensically reliable evidence and disrupt the overall incident analysis process. The first is that the sharing of physical resources among multiple users prevents their removal for analysis as this would violate the privacy of individuals not involved in the investigation. The second is that the location of the physical resource in a different geographic region from where the crime was committed may prevent the collection of evidence if no cooperation agreements are in place. These aspects, if not taken into consideration, may undermine the credibility of the evidence and decrease the likelihood of their acceptance in a legal process. This work, analyzes proposals found in the literature aimed at solving such challenges, discussing its limitations. It proposes a solution that covers collection-gathering, transportation and storage of evidence, aiming at overcoming existing limitations in the state of art. The proposed solution, called Dizang , provides a way of correlating evidence and its virtual origin, allowing the transport and storage of such data without affecting its credibility. For this, it proposes a mechanism to identify uniquely the source of the evidence, in order to preserve the relationship evidence-source even though the latter does not exist in the system under investigation. In summary, Dizang focuses on (1) the reproducibility of the collection process, (2)the establishment of a link between the evidence collected and its origin, (3) the preservation of jurisdiction and privacy of users not involved in the investigation and (4)the evidence's chain of custody.

LISTA DE ILUSTRAÇÕES

1 Modelo Automático de Aquisição de Dados Forenses 27

2 Virtuoso 29

3 *A Log Based Approach Model* 30

4 *Backup Approach Model* 31

5 *FoRensic Open Stack Tools* 33

6 *Digital Forensic Framework for Cloud Environment* 35

7 *Digital Forensic as a Service - Indexed Data* 36

8 Janela deslizante de coleta de evidência 43

9 Arquitetura geral da solução Dizang 45

10 Evolução do uso do espaço em disco com o Dizang 47

11 Tempo de cópia da memória de um contêiner 48

12 Tempo de transporte da evidência 49

13 Comando para cópia crua da memória do processo do contêiner . . . 50

14 Parte do arquivo **/proc/pid/numa_maps** ANTES da injeção 50

15 Parte do arquivo **/proc/pid/numa_maps** APÓS a injeção 51

16 Conteúdo da memória de **libexample.so** no formato [endereço]: [conteúdo] 51

LISTA DE TABELAS

1 Comparativo de soluções de coleta de informações de memória de máquinas em nuvem para análise forense 40

LISTA DE ABREVIATURAS E SIGLAS

VM	*Virtual Machines*
SO	Sistema Operacional
SaaS	*Software as a Service*
PaaS	*Platform as a Service*
IaaS	*Infrastructure as a Service*
FaaS	*Forensics as a Service*
LXC	*Linux Conteiners*
VMI	*Virtual Machine Introspection*
VMM	*Virtual Machine Manager*
CSP	*Cloud Service Provider*
GRR	*Google Rapid Response*
FROST	*FoRensic Open Stack Tools*
API	*Aplication Programing Interface*
SWGDE	*Scientific Working Group on Digital Evidence*
ETL	*Extract, Transform and Load*
EDIPM	*Enhanced Digital Investigation Process Model*
SENASP	Secretaria Nacional de Segurança Pública
NAS	*Network Accessed Storage*
GDB	*GNU Debugger*

SUMÁRIO

1 Introdução **11**

1.1 Motivação 12

1.2 Objetivos 12

1.3 Justificativa 13

1.4 Organização do documento 14

2 Fundamentação Teórica **15**

2.1 Nuvens computacionais e contêineres 15

2.2 Forense digital e seus desafios 18

2.2.1 Admissibilidade da evidência em processo legal. 18

2.2.2 Volume de dados para coleta 20

2.2.3 Privacidade e jurisdição 21

2.2.4 Coleta de evidências de memória volátil de máquinas em nuvem 22

2.3 Forense digital e gestão de incidentes 23

3 Revisão da literatura **25**

3.1 Soluções para análise forense em nuvem 25

3.1.1 Modelo automático de aquisição de dados para fins forenses . 25

3.1.2 Introspecção em máquina virtual 26

3.1.3 Virtuoso 28

3.1.4 Abordagem baseada em logs 28
3.1.5 Abordagem baseada em backups 30
3.1.6 Arcabouço forense para OpenStack 32
3.1.7 Forense como serviço 33
3.1.8 Abordagem de indexação de dados coletados para fins forenses 34
3.2 Aspectos relacionados a coleta de evidência 36
3.2.1 Acessar e coletar as informações de memória das máquinas virtuais em nuvem 36
3.2.2 Capacidade de reproduzir o processo e obter os mesmos resultados 37
3.2.3 Não violar privacidade ou jurisdição das partes não envolvidas na investigação 38
3.2.4 Garantir a cadeia de custódia da evidência 39
3.3 Resumo 39

4 Proposta de projeto: Dizang **41**

4.1 Identificação da origem 41
4.2 Descrever o sistema antes e depois do incidente 42
4.3 Garantindo integridade, confidencialidade e protegendo privacidade e jurisdição 44
4.4 Implementação 44
4.5 Resultados experimentais 46
4.5.1 Análise do desempenho 46

4.5.2 Identificação de injeção de código malicioso 49

4.6 Limitações 52

5 Conclusões e recomendações para trabalhos futuros 53

5.1 Conclusões 53

5.2 Trabalhos Futuros 54

Referências 55

1 INTRODUÇÃO

Técnicas de virtualização, replicação de serviços e compartilhamento de recursos entre múltiplos usuários (multi-inquilinato) proveem a nuvens computacionais uma elevada escalabilidade (MORSY; GRUNDY; MULLER, 2010). Ao mesmo tempo, tais mecanismos também criam grande 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 que a hospedam e balancear a carga entre elas. Isso permite ao sistema em nuvem atender à demanda dos usuários da aplicação 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 desperdício de recursos.

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, pode-se supor um cenário em que 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., em arquivos de *log*). Nesse caso, as evidências desse evento podem ser completamente perdidas após as instâncias em questão 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 pública ou híbrida (KEYUN et al., 2011). Especificamente, o multi-inquilinato dificulta a obtenção do *hardware* que executa as aplicações de interesse. Afinal, como ele é compartilhado por vários usuários, removê-los para análise poderia levar a uma violação de privaci-

dade dos usuários não relacionados à investigação. Já a natureza distribuída da nuvem pode levar à alocação de informações relevantes em vários países, dificultando a sua obtenção, em especial quando não existem acordos de cooperação entre as entidades envolvidas (DYKSTRA; SHERMAN, 2012). Combinadas, tais características limitam a coleta de evidências com a credibilidade necessária para que elas possam ser aceitas em processos legais (RAHMAN; KHAN, 2015).

1.1 Motivação

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 (DYKSTRA; SHERMAN, 2013) e (REICHERT; RICHARDS; YOSHIGOE, 2015) tratam de fatores como multi-inquilinato e multi-jurisdição, discutindo formas de coleta e preservação da evidência fora da nuvem. Já estudos como (GEORGE; VENTER; THOMAS, 2012) se concentram na análise forense para a coleta de evidência de máquinas virtuais em tempo de execução, enquanto trabalhos como (SANG, 2013) abordam processos de garantia da 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 máquina virtual) seja desalocado, haja condições de se reproduzir o processo de coleta de evidências.

1.2 Objetivos

O presente trabalho visa suplantar as limitações identificadas na literatura com relação à capacidade de coleta de evidências de aplicações em nuvem. Mais precisamente, por meio de uma proposta que tem como focos (1) a reprodutibilidade do

processo de coleta, (2) o estabelecimento de um 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. Desta forma, a solução buscada deve prover uma forma de obter dados relevantes para investigações ao mesmo tempo que permita transportar e armazenar tais dados preservando sua credibilidade. Para isso, a proposta supõe que o sistema sendo monitorado é executado dentro de um ambiente virtualizado em nuvem. A presente proposta tem como foco prover uma solução técnica que atenda as necessidades jurídicas presentes hoje, não faz parte dos objetivos deste trabalho propor alterações na legislação ou nos procedimentos legais vigentes.

1.3 Justificativa

O crescente volume de informações que as soluções em nuvem armazenam e trafegam atualmente, bem como os aspectos de multi-inquilinato e a multi-jurisdição dos provedores de nuvem, estão entre os principais obstáculos enfrentados pelos investigadores forenses. (QUICK; CHOO, 2014) (KEYUN et al., 2011). Desta forma, a criação de soluções que permitam melhorar o processo de investigação forense nesse cenário tem relevância não apenas teórica, mas também prática.

Ao mesmo tempo, virtualização tem sido amplamente adotada por empresas das mais diversas áreas. Segundo o ”State of the Cloud Report”, relatório produzido pela empresa Right Scale (Right Scale, 2018), 96% das organizações entrevistadas estão utilizando ou experimentando soluções em nuvem no modelo IaaS (*Infrastructure as a Service* – Infraestrutura como um Serviço). Embora a virtualização de recursos na nuvem seja tradicionalmente feita por meio de máquinas virtuais, tem ganhado importância também a tecnologia de contêineres (BERNSTEIN, 2014), uma forma de virtualização bastante leve e flexível. De fato, conforme o “Container Market Adoption Survey 2018”, realizado com 519 empresas que têm desenvolvimento de software como sua atividade fim ou como suporte à atividade fim, 83,5% dos respondentes uti-

lizam contêineres em ambiente produtivo em suas arquiteturas de microsserviços em nuvem.

Por fim, gestão de incidentes, que cada vez mais compartilha processos em comum com forense digital, pode se beneficiar de uma solução de coleta e preservação de evidências que esteja pronta na fase de *preparação ao incidente* e seja capaz de preservar as evidências para a fase de *pós-incidente* permitindo assim que na fase de *detecção e análise* não seja preciso se preocupar com perda parcial ou total de evidências.

Essa tendência motiva a construção de soluções que levem em consideração as particularidades inerentes aos recursos disponíveis em nuvens computacionais.

1.4 Organização do documento

O restante deste documento está organizado conforme os seguintes capítulos.

O Capítulo 2 apresenta o ambiente em que este trabalho está inserido, as tecnologias envolvidas e os desafios a serem superados.

O Capítulo 3 analisa os trabalhos relacionados a forense de memória em nuvem.

O Capítulo 4 apresenta as atividades realizadas como parte do trabalho de pesquisa aqui apresentado, bem como suas conclusões.

2 FUNDAMENTAÇÃO TEÓRICA

Desde 2001, diversos modelos para a condução de investigação digital foram propostos. O *Enhanced Digital Investigation Process Model*, proposto por Carrier e Stafford em 2003, é a última iteração na evolução do processo forense digital (GRISPOS; STORER; GLISSON, 2012). Entretanto, como tais modelos de investigação foram desenvolvidos antes da aparição de tecnologias de computação em nuvem, muitos partem do pressuposto que o investigador tem acesso e controle sobre o sistema sob investigação (GRISPOS; STORER; GLISSON, 2012). Esta defasagem é um dos desafios da forense digital atual.

Neste capítulo, são discutidos os principais conceitos que permeiam esse cenário. Especificamente, após uma discussão geral sobre computação em nuvem e suas particularidades, são discutidas os características esperadas de um processo de forense digital robusto e sua interseção com gestão de incidentes.

2.1 Nuvens computacionais 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. Existem três modelos principais de comercialização de uso da nuvem (MELL; GRANCE, 2011):

- SaaS (*Software as a Service* – *Software* como serviço): Onde se provê o *software* que será utilizado; nesse caso, os clientes do serviço são os usuários finais do software.

- PaaS (*Platform as a Service* – Plataforma como serviço): Onde se provê o ambiente para o desenvolvimento, teste e execução do *software*; nesse caso, os clientes do serviço são desenvolvedores de aplicações.

- IaaS (*Infrastructure as a Service* – Infraestrutura como serviço): Onde são fornecidos recursos computacionais básicos, como processamento, memória e redes, em geral de forma virtualizada; os clientes desse tipo de serviço costumam ser arquitetos de sistemas.

O tipo de serviço de nuvem mais pertinente para este trabalho é o IaaS, uma solução muito usada atualmente pela sua capacidade de prover recursos sob demanda de forma auto-escalável. Nesse cenário, o uso intenso de tecnologias de virtualização costuma levar a recursos altamente voláteis, que são alocados e desalocados a qualquer momento pelo orquestrador da nuvem para suprir eventuais aumentos e reduções de demanda. É possível até mesmo construir scripts para a automatizar a construção da arquitetura desejada, permitindo a instanciação e interconexão de máquinas adequadas para a atividade fim do sistema. Esses scripts, escritos em linguagem específica como Puppet (Puppet,), Chef (Chef,) ou Vagrant (Vagrant,), podem conter instruções de como distribuir o tráfego de rede entre as diferentes instâncias de computação ou armazenamento. Dentre as vantagens de sua utilização, podem ser citadas a capacidade de usar os recursos de nuvem de uma forma mais eficiente, levar a uma menor necessidade de intervenção humana, e prover maior resiliência a variações de demanda do sistema.

Uma tecnologia de virtualização possível para cenários de nuvem, e cuja utilização vem crescendo nos últimos anos, são os chamados contêineres (BERNSTEIN, 2014).

Basicamente, um contêiner é um método de virtualização do sistema operacional que permite executar uma aplicação, bem como suas dependências, em um processo no qual recursos como disco, memória e rede permanecem isolados. Diferente das máquinas virtuais, a virtualização com contêineres é feita no nível do Sistema Operacional (SO) nativo. Como resultado, tem-se uma implementação de virtualização na qual eliminam-se camadas entre o aplicativo executado e o *hardware* físico, permitindo maior granularidade no controle sobre esses recursos e melhorando a eficiência da infraestrutura.

Uma implementação bastante utilizada para esse propósito são os LXC (*LinuX Conteiners* – Contêineres Linux) (LINUXCONTAINERS.ORG, 2015), que aproveitam-se de funcionalidades como *cgroups*, *kernel namespacing* e *chroot* do kernel do Linux para auxiliar no gerenciamento e isolamento de recursos virtuais. Mais precisamente, a funcionalidade de *cgroups* (*Control Groups* – Grupos de Controle) presente no Kernel do Linux limita e isola o uso de recursos como CPU, memória e disco de um conjunto de processos, além de organizá-los de forma hierárquica. O trabalho nessa funcionalidade começou em 2006 na Google sob a denominação de *process container*. No final de 2007, seu nome foi alterado para *control groups*, e o resultado foi então adicionado à versão 2.6.24 do kernel lançado em 2008 (Unix Man Pages, a).

Já o *Namespacing* é uma funcionalidade do Kernel do Linux usada para isolar e virtualizar recursos do sistema operacional, como identificadores de processos, acessos à rede, comunicação inter-processos e sistema de arquivos. *Namespacing* envolve os recursos do sistema operacional em uma abstração que faz parecer aos processos de um mesmo *namespace* que eles tem sua própria instância isolada de um recurso global. Desta forma, essa é a principal funcionalidade por trás da implementação de Contêineres Linux (Unix Man Pages, d).

Finalmente, *chroot* (*Change Root* – Mude Root) é uma funcionalidade do Kernel do Linux usada para mudar o diretório *root* enxergado pelo processo que está chamando a função, bem como por todos os seus processos filhos. A chamada a *chroot* altera o processo de resolução de caminhos do sistema operacional para o processo que o chamou (Unix Man Pages, b). Desta forma, pode-se instalar uma distribuição Linux secundária em uma pasta, ao invés de uma partição, e executar programas desta pasta sem perda significativa de desempenho.

2.2 Forense digital e seus desafios

A área de forense digital (também conhecida por forense computacional) refere-se a um conjunto de técnicas de coleta e análise da interação entre humanos e computadores de forma que suas conclusões sejam aceitas em um processo legal. Tal como a forense tradicional, a forense digital se baseia no princípio de Locard, estabelecido pelo médico francês Edmond Locard da seguinte forma: "Quando um indivíduo entra em contato com outro objeto ou indivíduo, este sempre deixa vestígio deste contato" (RAMOS, 2011). De forma similar, a forense digital tem por objetivo a investigação de evidências digitais da interação entre homem e máquina, de modo a reconstruir a cadeia de eventos passados para que suas conclusões sejam validadas por terceiros e sejam aceitas em um processo legal.

A seguir são detalhados os principais desafios enfrentados pela forense digital quando aplicada a infraestruturas em nuvem.

2.2.1 Admissibilidade da evidência em processo legal.

O processo de análise forense no evento de um crime digital é descrito no EDIPM (*Enhanced Digital Investigation Process Model* – Modelo Melhorado para Processo de Investigação Digital) na forma de 4 fases (GRISPOS; STORER; GLISSON, 2012):

identificar, preservar, examinar e apresentar. A fase mais pertinente a este trabalho é a de preservação da evidência, que deve ser conduzida de forma forensicamente aceitável. Ou seja, deve-se coletar as evidências de forma que elas sejam aceitas em um processo legal e não tenham sua credibilidade questionada no curso do mesmo.

Para atingir tal objetivo, o primeiro passo é a garantia da cadeia de custódia relacionada a evidência. Cadeia de custódia é o processo de documentação da história cronológica da evidência de modo a saber onde a evidência esteve e quem teve acesso a ela (RAMOS, 2011). Uma cadeia de custódia idealmente deve tornar visível o estado da evidência antes, durante e após a interação com o processo de investigação (LUIS; SANCHEZ; GIOVA, 2016). A razão para essa preocupação é que alguns processos investigativos podem causar alteração da evidência na forma que foi coletada, como, por exemplo, a consolidação em um único local dos arquivos que compõem uma base de dados distribuída.

O passo seguinte é a garantia da autenticidade e da integridade da evidência. Autenticidade pode ser definida como "o processo pelo qual se pode garantir a autoria do documento eletrônico" (RAMOS, 2011), ou seja, por meio do qual se não permite dúvida quanto à identificação do autor . Já a integridade pode ser definida como "o atestado da inteireza do documento eletrônico após sua transmissão, bem como apontar eventual alteração irregular de seu conteúdo" (RAMOS, 2011). Caso haja dúvida acerca de um desses requisitos, uma perícia técnica pode ser convocada. Nesse caso, durante a perícia é analisado o autor da evidência, ou seja, verifica-se sua fonte e se a mesma não foi alterada no processo.

Em uma infraestrutura física, a coleta de evidências pode ser feita de forma relativamente simples, bastando-se remover o recurso físico, transportar este para um laboratório e lá analisar os dados. Para limitar a exposição da evidência a manipulações indevidas, ela pode ser mantida em uma sala-cofre, à qual o acesso é controlado. A reprodutibilidade do processo de coleta e a manutenção da integridade da evidência

são, então, tarefas bem diretas.

Já em um cenário de computação em nuvem, especialmente as de infraestrutura auto-escalável, existe um conjunto de novos desafios. Primeiramente, em contraste com infraestruturas tradicionais, o recurso físico em princípio não pode ser removido: como os recursos são utilizados por outros usuários não relacionados à investigação, fazê-lo constituiria violação de privacidade. A volatilidade dos recursos também torna a verificação do seu autor um processo mais complexo, pois o recurso que gera certa evidência pode deixar de existir algum tempo depois de fazê-lo (SIMOU et al., 2014). A integridade da evidência também acaba sendo uma tarefa não trivial, pois ela precisa ser coletada, transportada e armazenada, o que caracteriza a necessidade de preservação da cadeia de custódia. A violação de qualquer uma dessas características pode colocar em dúvida a credibilidade da evidência. Embora a admissibilidade da evidência em um processo legal seja uma decisão do juiz a credibilidade da mesma tem papel importante nesta decisão.

2.2.2 Volume de dados para coleta

O processo de coleta de evidências na forense digital herda suas práticas da forense tradicional, na qual isola-se cena do crime e coletam-se as evidências presentes. Transportando esse método para a forense digital, introduz-se a realização da cópia bit a bit da informação que se deseja analisar. No passado, com as soluções manipulando quantidades bem menores de memória, disco e tráfego, tal prática não era considerada muito problemática. Entretanto, nas atuais soluções, aplicações e arquiteturas em nuvem, o volume de dados é consideravelmente maior (QUICK; CHOO, 2014). Afinal, o acesso fácil e dinâmico a recursos de armazenamento e de processamento, como máquinas virtuais, balanceadores de carga e *firewalls*, acaba também aumentando a quantidade elementos geradores de evidência. Esse problema é ilustrado em (QUICK; CHOO, 2014), onde se discute uma investigação na qual a quantidade de dados a serem

analisados para fins de forense digital tomaria 6 meses.

Encontrar uma forma de armazenar menos informações e, assim, tornar a fase de análise mais rápida e eficiente, é um passo importante para garantir a celeridade de investigações forenses.

2.2.3 Privacidade e jurisdição

A prática de remoção de equipamentos para coleta de evidências, mencionada na seção 2.2.2, também traz consequências negativas do ponto de vista de privacidade. Mais precisamente, em soluções envolvendo infraestruturas físicas, tal prática não costuma trazer grandes problemas porque os objetos ou indivíduos sob investigação normalmente estão diretamente relacionados ao equipamento removido. Nas soluções em nuvem, entretanto, tal prática não é recomendada porque o recurso físico é compartilhado por vários usuários, inclusive indivíduos não envolvidos na investigação. Logo, remover tais recursos configuraria violação de privacidade. Como um complicador adicional, o fato de os dados potencialmente não estarem armazenados no mesmo território em que a investigação é realizada acaba demandando acordos de cooperação jurídica entre as partes, o que nem sempre é possível (SIMOU et al., 2014).

O ”State of the Cloud Report”, relatório produzido pela empresa Right Scale (**??**), cita nas páginas 21 e 22 que os maiores desafios na adoção de nuvem são, *redução de custo* e *segurança*. Embora privacidade e jurisdição sejam aspectos distintos e possuidores de suas particularidades, neste documento eles são discutidos em conjunto pois em busca de otimização de custos as empresas tendem a buscar regiões mais baratas e compartilhamento do recurso físico.

Neste cenário, encontrar uma forma de coletar a evidência sem violar jurisdição e privacidade ganham importância.

2.2.4 Coleta de evidências de memória volátil de máquinas em nuvem

A prática de armazenar histórico de tráfego de rede e alterações de dados armazenados em disco já é bem difundida na comunidade forense. Por outro lado, a memória volátil de computadores não costuma receber o mesmo tratamento: suas alterações quase nunca são armazenadas, seja por questões de desempenho ou por simples praticidade, dado a reduzida sobrevida dessas informações. Infelizmente, isso acaba dificultando a análise de uma classe específica de ataques, conhecidos como injeção de código em memória (CASE et al., 2014). Assim, quando usados contra uma arquitetura em nuvem, tais ataques não deixam rastros quando recursos de processamento virtuais são desativados e sua memória é liberada (VöMEL; STüTTGEN, 2013; CASE et al., 2014). Em particular, têm especial interesse quatro tipos particulares dessa família de ameaças (CASE et al., 2014):

- **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. Tal estratégia, comumente usada para instalar códigos malicioso, pode fazer com que uma biblioteca maliciosa armazenada no sistema seja distribuída por vários processos de uma mesma máquina, dificultando sua remoção (MILLER; TURKULAINEN, 2004).

- **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 (FEWER, 2008).

- **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.

2.3 Forense digital e gestão de incidentes

Gestão de incidentes é uma estratégia para reduzir os riscos à confidencialidade, integridade e disponibilidade de ativos de uma organização, bem como minimizar sua perda. As fases relevantes de uma estratégia de gestão de incidentes são (Ab Rahman; CHOO, 2015):

- **Preparação**: Onde se organiza o ambiente para minimizar impactos de um incidente, nesta fase também é definido o time de resposta a incidente que tem como responsabilidade determinar o que ocorreu e quais ações devem ser tomadas.

- **Detecção**: Tem como objetivo minimizar o risco das ameaças que não foram antecipadas. A fase de detecção começa quando alguma atividade suspeita é detectada por algum serviço automático como um sistema de detecção de ameaças ou por intervenção humana como o de um responsável pela análise de risco.

- **Resposta ao incidente**: Contém as fases de *contenção*, *erradicação* e *recuperação*. Não existe um procedimento que atenda a todas os cenários, os procedimentos são feitos de acordo com a natureza e as necessidades do negócio.

- **Pós incidente**: Última fase da gestão de incidentes, sua principal preocupação

é coleta de informações das três fases anteriores para alimentar um processo de aprendizado visando evitar futuros incidentes semelhantes.

A gestão de incidentes tem o foco em conter falhas de segurança. Considerações como coleta de evidências costuma ser secundário (Ab Rahman; CHOO, 2015). A oportunidade de se aplicar processos de coleta forenses em gestão de incidente já foi levantado por (RAHMAN; CAHYANI; CHOO, 2016) uma vez que ambos compartilhar de um ferramental em comum.

Ferramental e técnicas forenses estão se tornando úteis não só para geração de evidências para processos jurídicos mas também para ajudar na reconstrução de eventos. Incorporando práticas forenses na estratégia de gestão de incidente ajuda a gerar as evidências necessárias para eventual processo jurídico como também proteger as evidencias de dano como resultado das fase de resposta ao incidente.

3 REVISÃO DA LITERATURA

Os principais aspectos relacionados ao tratamento de evidências para análise forense em nuvem são: coleta, transporte, armazenamento, garantia da cadeia de custódia e reprodutibilidade do processo de coleta. Este capítulo descreve o estado da arte e discute como a presente proposta se insere nesse contexto.

3.1 Soluções para análise forense em nuvem

Nesta seção são descritas algumas das principais propostas existentes para coleta de evidências forenses, fornecendo uma visão geral do estado da arte na área.

3.1.1 Modelo automático de aquisição de dados para fins forenses

O modelo proposto por (REICHERT; RICHARDS; YOSHIGOE, 2015), ilustrado na Figura 1, é um processo de coleta de evidências integrado ao *hypervisor* e disparado por algum sistema de detecção de intrusão. A partir do momento em que uma ameaça é detectada, o modelo dita que sejam criados instantâneos (*snapshots*) das máquinas virtuais comprometidas. O intervalo de tempo em que os instantâneos de memória são gerados é configurável, e todos eles são armazenados em persistência. O modelo se preocupa em, de forma automatizada, excluir informações de clientes não relacionados à investigação e também em armazenar o restante em local seguro. O autor não descreve os detalhes do armazenamento, embora afirma que isso deve ser feito de forma "forensicamente aceitável".

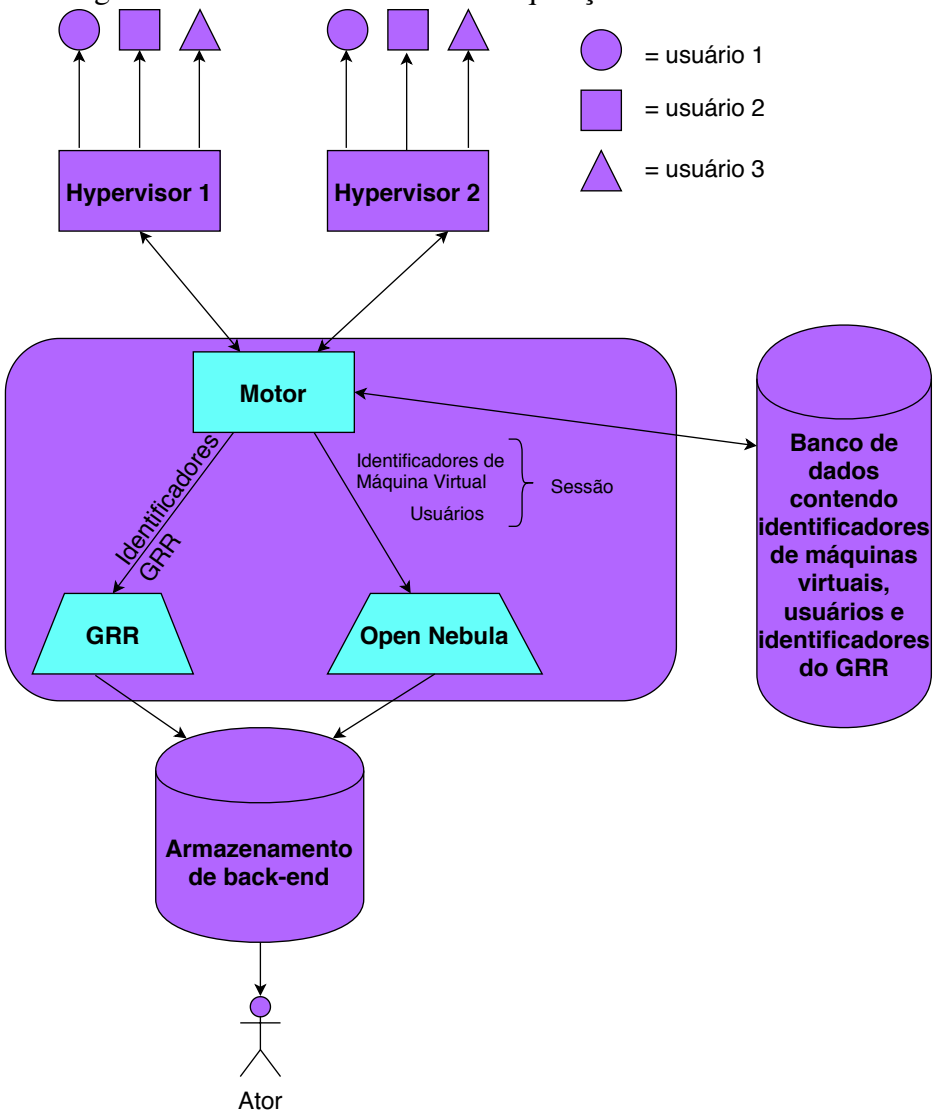
Para agregar e analisar as evidências coletadas, o modelo faz uso do GRR (*Google Rapid Response* – Resposta Rápida Google), uma ferramenta de resposta a incidentes criado no projeto 20% na Google para facilitar análises forenses de forma remota (e.g., acesso de baixo nível ao disco e à memória das máquinas analisadas) (GOOGLE, 2013). O tratamento de evidências usa um motor de regras baseado em um conjunto de descrições de ameaças conhecidas, as quais são armazenadas em um banco de dados. Caso alguma evidência coletada coincida com ameaças armazenadas, esse motor alerta um usuário humano para uma avaliação mais detalhada. Este modelo tem dentre suas principais vantagens a automação do processo de coleta, já que, ao menos em parte, ele dispensa intervenção humana.

Quando acionado, o Motor gera instantâneos das máquinas virtuais administradas por *Hypervisor 1* e *Hypervisor 2* e os envia para o Armazenamento de *Back-End*. Os instantâneos são instalados em um ambiente controlado junto com agentes GRR que extraem os dados para análise. Estes dados são então comparados com ameaças já conhecidas pelo modelo.

3.1.2 Introspecção em máquina virtual

A proposta de (POISEL; MALZER; TJOA, 2013) é baseada na técnica de VMI (*Virtual Machine Introspection* – Introspecção em Máquina Virtual) para coleta de memória volátil. Essa técnica se apoia no fato de que o VMM (*Virtual Machine Manager* – Gerenciador de Máquinas Virtuais) mapeia os recursos alocados para máquinas virtuais nos recursos físicos correspondentes da máquina hospedeira. Este mapeamento é usado para permitir que a memória volátil copiada da máquina virtual seja reconstruída em uma máquina física, para análise posterior. A proposta realiza coleta contínua de instantâneos de memória durante o funcionamento do sistema, sem distinção do que aconteceu antes ou depois do fato de interesse, e todos os instantâneos de memória são armazenados para análise. Visando eliminar a chance de inconsistências no ins-

Figura 1: Modelo Automático de Aquisição de Dados Forenses



Adaptado de (REICHERT; RICHARDS; YOSHIGOE, 2015)

tantâneo de memória volátil, a máquina virtual tem sua execução suspensa durante o processo de extração.

Uma desvantagem da técnica de VMI, mencionada pelo próprio autor em (POISEL; MALZER; TJOA, 2013), no capítulo 3.1, é a necessidade de tradução de endereços de memória da máquina virtual em endereços de memória da máquina física hospedeira. Como essa tradução depende de conhecimento do que está sendo executado na máquina virtual, uma solução baseada em VMI não é completamente portável,

sendo necessárias adequações para diferentes clientes. Além disso, esta tradução de endereços pode ser computacionalmente custosa.

3.1.3 Virtuoso

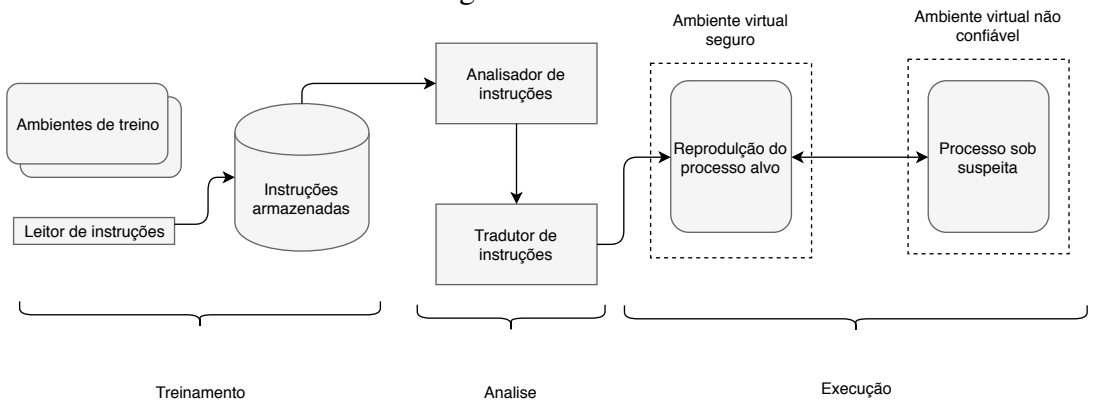
Também na vertente de introspecção de máquina virtual, (DOLAN-GAVITT et al., 2011) propõe o *Virtuoso*, um arcabouço de coleta de informações de processos específicos em uma máquina virtual. O arcabouço funciona em três fases. A primeira realiza um estudo em uma máquina virtual de testes, mapeando o conjunto de instruções executado pelo processo do qual se deseja coletar dados de memória. O estudo é realizado no Ambientes de Treino e o Leitor de Instruções coleta e armazena as instruções geradas pelo processo alvo da análise e os armazena no banco de dados de Instruções Armazenadas. Na segunda fase, o Analisador de Instruções cria um executável a partir do conjunto de instruções coletado na fase anterior. Estas instruções precisam ter suas referências de memória traduzidas para que seja possível executá-las fora do ambiente original, o que ocorre no Tradutor de Instruções. Com o executável, a terceira fase usa um Ambiente Virtual Seguro na máquina hospedeira capaz de acessar os endereços de memória do Ambiente Virtual Não Confiável, tornando possível coletar instantâneos de memória do processo em execução. A Figura 2 ilustra o funcionamento desse arcabouço.

A principal vantagem do Virtuoso é a capacidade de coletar instantâneos de memória de um processo específico e reproduzi-lo em um ambiente confiável. Uma desvantagem importante, entretanto, é que sua atuação se dá apenas após o evento que se deseja avaliar forensicamente.

3.1.4 Abordagem baseada em logs

O arcabouço proposto por (SANG, 2013) é um sistema que funciona em parceria com o provedor de nuvem: o provedor último envia informações ao arcabouço, que

Figura 2: Virtuoso

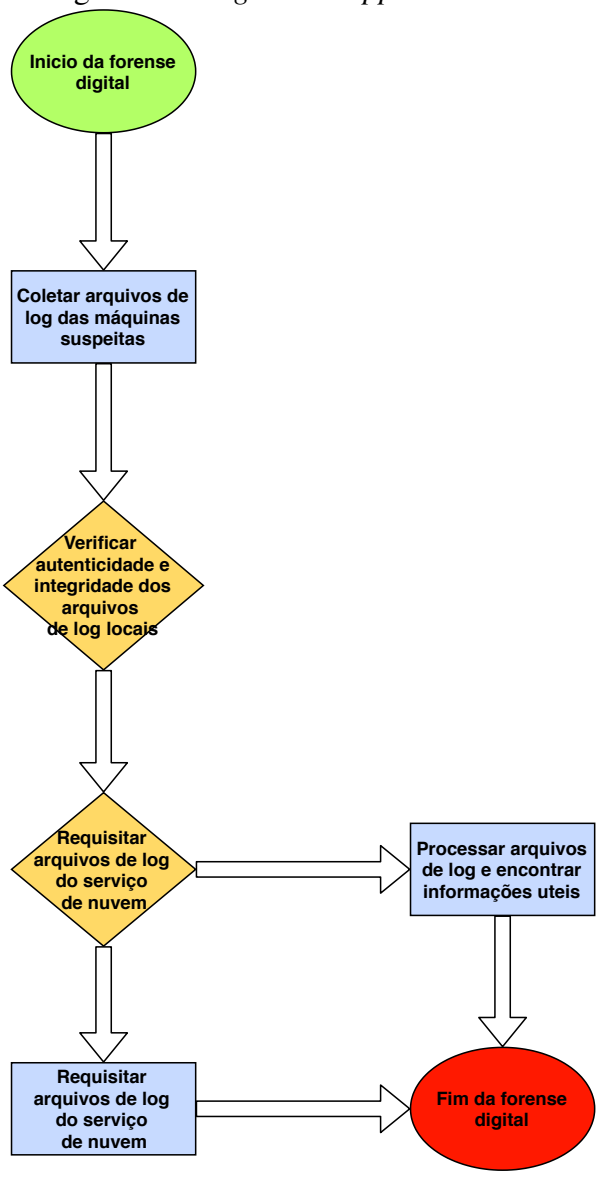


Adaptado de (DOLAN-GAVITT et al., 2011)

por sua vez as armazena em um local adequado, de forma centralizada. O conjunto de informações armazenadas é negociado antecipadamente com o provedor de nuvem, indo desde instantâneos de memória volátil até pacotes trafegados nas interfaces de rede da máquina virtual. O arcabouço coleta informações continuamente e usa cálculo de hash das evidências enviadas pelo provedor de nuvem para garantir que elas não foram alteradas durante o transporte. A Figura 3 ilustra o funcionamento da solução, focando em um caso específico de log de rede, de modo similar ao descrito em (SANG, 2013).

Assim como as propostas anteriores, o arcabouço em questão também não faz distinção do que aconteceu antes ou depois do fato de interesse, mas coleta constantemente informações da máquina virtual. Outra potencial limitação é que o arcabouço depende da cooperação do provedor de nuvem. Tal dependência é uma estratégia considerada fraca pela comunidade forense, pois a prioridade do CSP (*Cloud Service Provider* – Provedor de Serviços de Nuvem) é a de garantir a disponibilidade do serviço, não de coletar evidências (ALQAHTANY et al., 2015).

Figura 3: *A Log Based Approach Model*



Adaptado de (SANG, 2013)

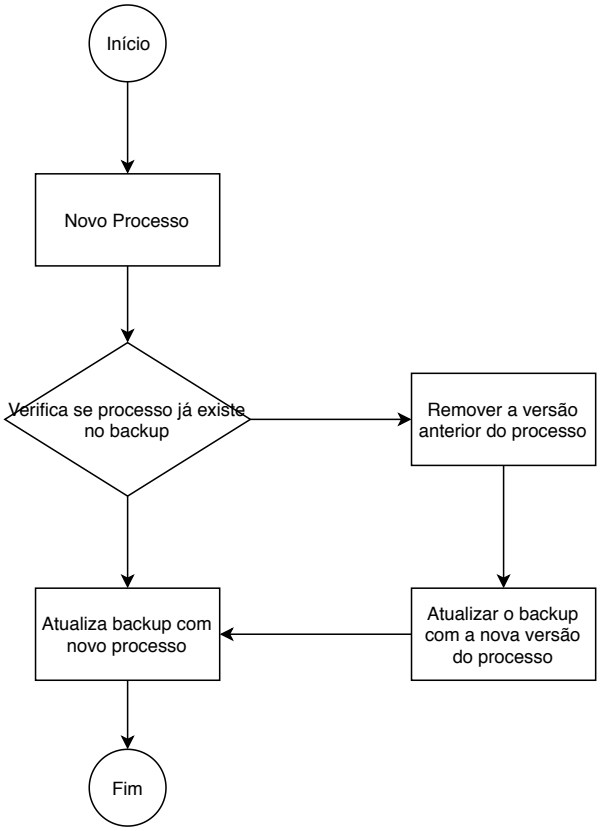
3.1.5 Abordagem baseada em backups

O trabalho descrito em (DEZFOULI et al., 2012) é voltado a dispositivos móveis e tem como principal característica a preocupação com as limitações de armazenamento do dispositivo. Por essa razão, o processo de coleta de instantâneos de memória volátil e armazena separadamente as informações de cada processo que está ativo de modo a permitir um gerenciamento adequado do espaço em disco. A solução também se

preocupa em descartar informações de processos que foram terminados e removidos da memória, além de buscar o uso consciente do espaço de armazenamento disponível no dispositivo. A Figura 4 mostra, em alto nível, a forma como o armazenamento de evidências é gerenciado.

Como em diversas outras propostas, processo de coleta de informações de memória volátil em (DEZFOULI et al., 2012) é executado continuamente, independente de eventos de interesse (e.g., detecção de ameaças). Um outro fator que conta como desvantagem nesta proposta é o processo não armazenar histórico das coletas anteriores, como é possível ver na Figura 4. A coleta anterior é substituída pela nova.

Figura 4: *Backup Approach Model*



Adaptado de (DEZFOULI et al., 2012)

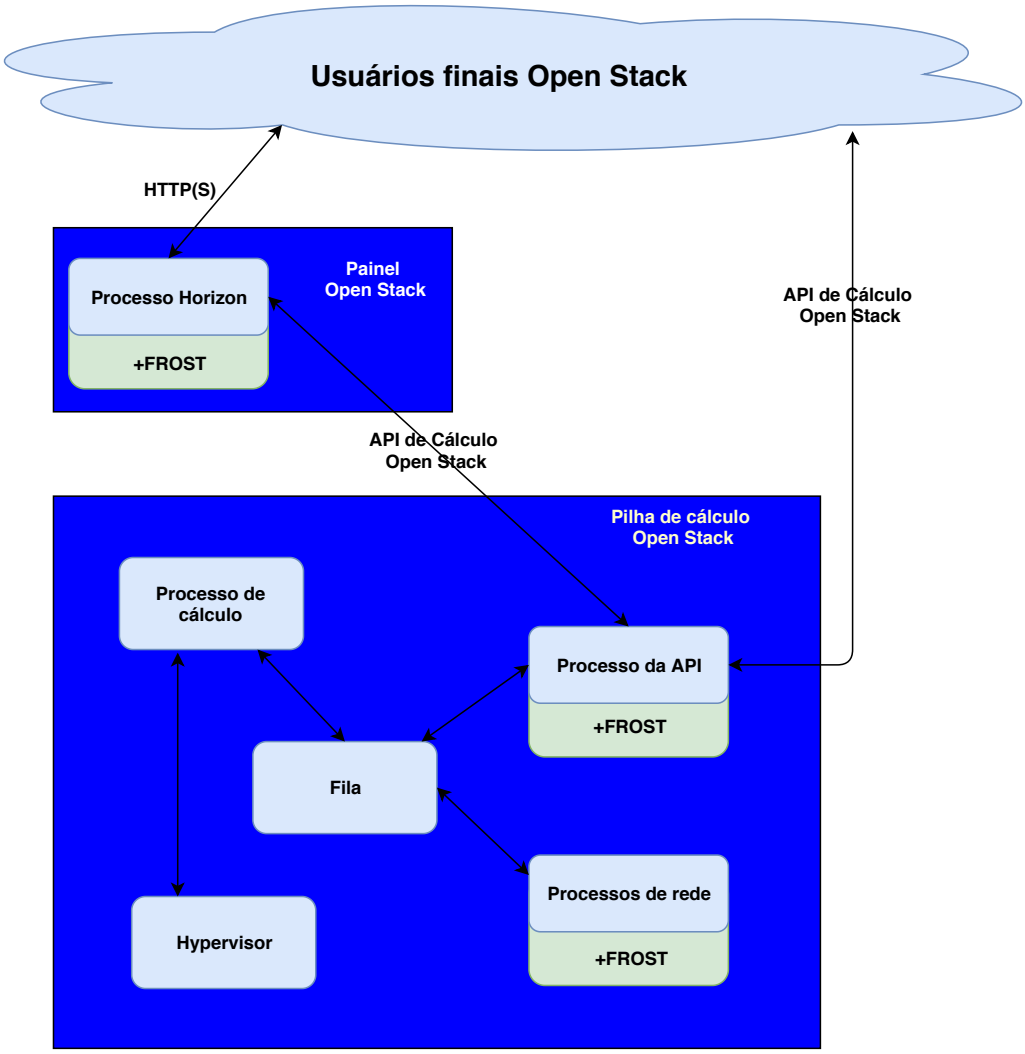
3.1.6 Arcabouço forense para OpenStack

A ferramenta FROST (*FoRensic Open Stack Tools* – Ferramentas Forenses para Arcabouço Open Stack), proposta por (DYKSTRA; SHERMAN, 2013), consiste em um conjunto de bibliotecas integradas ao OpenStack, um dos arcabouços de gerenciamento de infra estruturas virtualizadas bastante difundido (STACK, 2018). Por meio dessa integração, o FROST expõe um conjunto de APIs que podem ser usadas por aplicações de coleta de evidências forenses. Essas APIs dão acesso a recursos da máquina virtual administrada, tais como disco, logs de tráfego de rede e memória volátil. A proposta descreve apenas o arcabouço, deixando a critério do usuário detalhes como periodicidade e tamanho da coleta, bem como a forma de transporte da evidência e onde ela é armazenada.

A Figura 5 ilustra a integração entre FROST e o arcabouço *OpenStack*. Nela é possível ver os dois pontos desta integração. O primeiro em sua camada de interface de administração web onde o usuário pode acionar a coleta de artefatos para análise. A segunda ocorre em seu núcleo, adiciona novas chamadas a API do *Open Stack* para viabilizar a coleta de informações da máquina virtual e também integra com os processos de rede para extrair logs de tráfego da rede.

De todas as propostas avaliadas, a FROST é a única que mostra preocupação com adequação a questões legais e padrões já estabelecidos na indústria forense. Especificamente, o autor declara que FROST segue as práticas definidas no SWGDE (*Scientific Working Group on Digital Evidence* – Grupo de Pesquisa Cientifica Em Evidência Digital) e do Manual de Busca e Apreensão do Departamento de Justiça Norte-Americano.

Figura 5: *FoRensic Open Stack Tools*



Adaptado de (DYKSTRA; SHERMAN, 2013)

3.1.7 Forense como serviço

O trabalho descrito em (GEORGE; VENTER; THOMAS, 2012) se concentra em monitoramento de rede, operando em uma arquitetura FaaS (*Forense as a Service* – Forense Como Serviço). Conforme ilustrado na Figura 6, a arquitetura da solução consiste em um conjunto de ferramentas com capacidade de descobrir automaticamente as interfaces sob monitoramento, além de coletar evidências de tais máquinas e armazená-las.

O processo de autodescoberta e associação das evidências com usuários de rede é realizado por um motor baseado em ontologias armazenadas em um banco de dados próprio. As ontologias são utilizadas para determinar a associação entre as evidências coletadas e os usuários ou artefatos que as geraram. O Minerador de dados de Log implementa um algoritmo de descoberta de arquivos de log de artefatos de rede definidos como relevantes. O Monitor de rede é responsável por interceptar o tráfego entre as partes consideradas suspeitas e por fim, a Monitoração do Sistema em Tempo Real captura informações específicas do sistema em execução como instantâneos dos processos sob investigação.

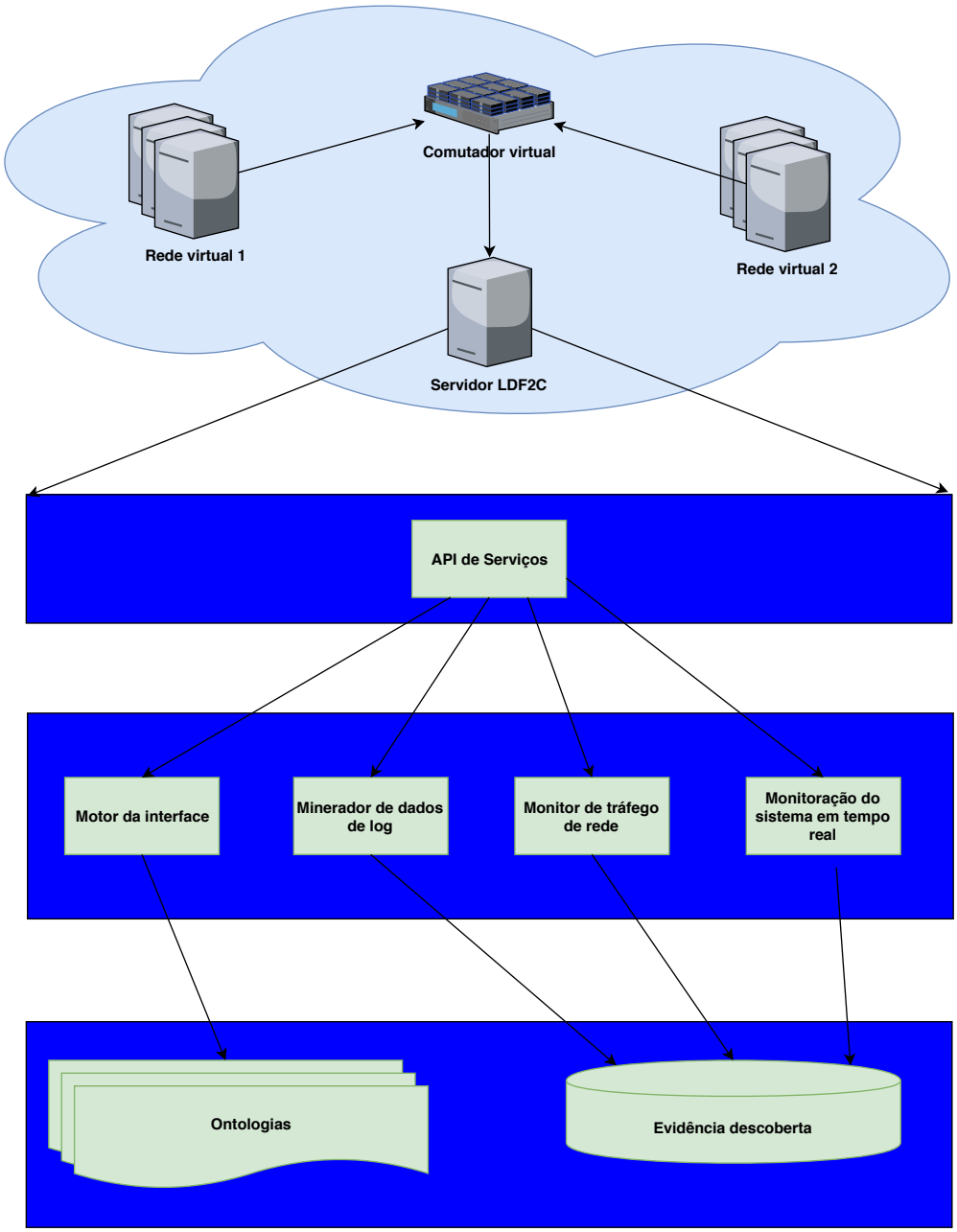
Entretanto, a proposta se concentra apenas no processo de coleta, enquanto a descrição dos mecanismos de armazenamento e transporte não é detalhada.

3.1.8 Abordagem de indexação de dados coletados para fins forenses

Na mesma vertente da solução de forense como serviço descrita na Seção 3.1.7, (LEE; UN, 2012) trata o problema de grande volume de dados coletados por meio de um serviço de coleta e indexação de evidências. O serviço espera receber dados da execução do comando unix DD (Unix Man Pages, c) nas máquinas alvo, onde, apoiado em processos ETL (*Extract, Transform and Load* – Extrair, Transformar e Carregar) e MapReduce (WIKIPEDIA, 2018), os dados são disponibilizados para consulta pelos investigadores. A coleta ocorre continuamente, em intervalos de tempo configuráveis.

A Figura 7 mostra a arquitetura da solução. Um NAS (*Network Accessed Storage* – Armazenamento acessado via rede) é usado para armazenar as evidências coletadas. Antes da análise dos dados pelos processos de MapReduce é necessário executar o processo de ETL para adequar a informação a necessidade específica da investigação. Esta fase ocorre nos Nós Filtrando. Em seguida a informação é submetida ao processo de MapReduce e armazenado em um HBase (HBase,). O Nó Mestre tem o papel de

Figura 6: *Digital Forensic Framework for Cloud Environment*



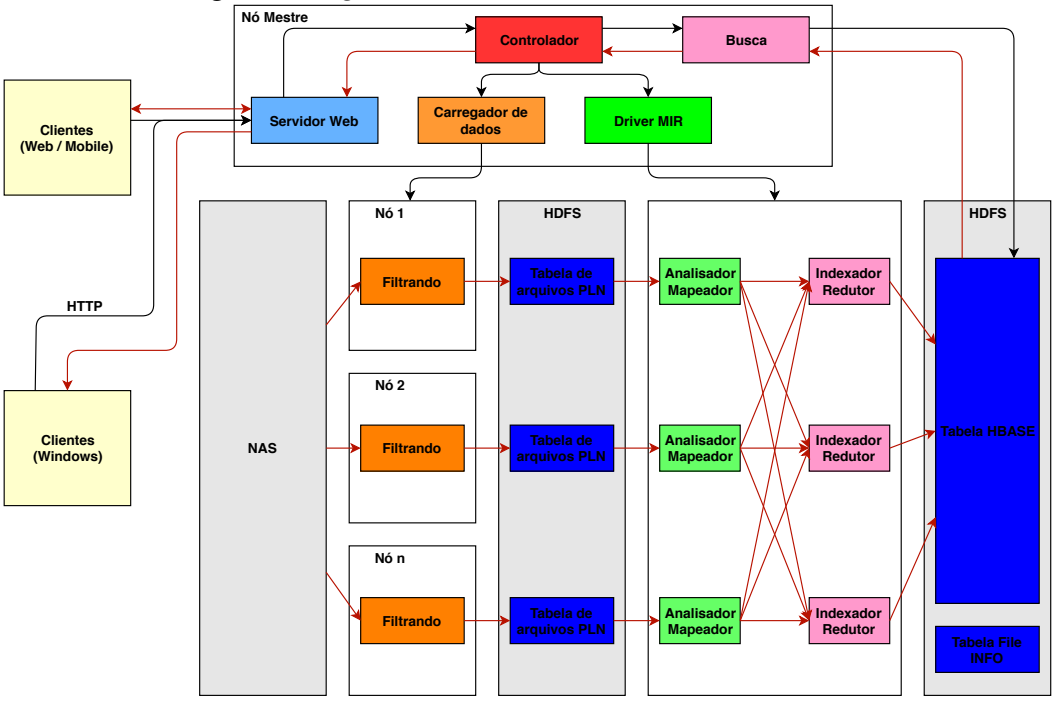
Adaptado de (GEORGE; VENTER; THOMAS, 2012)

orquestrador enviando os comandos de indexação dos dados coletados e recebendo as requisições de busca dos usuários via seu Servidor Web.

Embora interessante, a solução não deixa claro a localização do armazenamento dos dados coletados, nem quem é responsável pela infraestrutura de armazenamento.

Também não é discutido como os dados são transportados até o ponto de armazenamento, nem como garantir que esses dados não sejam alteramos no processo.

Figura 7: *Digital Forensic as a Service - Indexed Data*



Adaptado de (LEE; UN, 2012)

3.2 Aspectos relacionados a coleta de evidência

Para uma discussão mais estruturada, a seguir os trabalhos mencionados na Seção 3.1.2 são agrupados e avaliados com base nos diferentes aspectos que abordam.

3.2.1 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 (REICHERT; RICHARDS; YOSHIGOE, 2015; POISEL; MALZER; TJOA, 2013; DYKSTRA; SHERMAN, 2013; GEORGE; VENTER; THOMAS, 2012; SANG, 2013). 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 (CASE et al., 2014). Entre os trabalhos estudados, a única proposta encontrada que leva tal necessidade em consideração é (DEZFOULI et al., 2012), 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 (POISEL; MALZER; TJOA, 2013; DYKSTRA; SHERMAN, 2013; SANG, 2013; DOLAN-GAVITT et al., 2011), 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 disso, elas não tratam o cenário em que é necessário coletar evidências quando são liberados os recursos virtuais que as contêm.

3.2.2 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. Afinal, todas elas dependem da existência do recurso virtual para a repetição do processo de coleta.

3.2.3 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. A razão é 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 de duas estratégias principais: a primeira, adotada em (REICHERT; RICHARDS; YOSHIGOE, 2015; GEORGE; VENTER; THOMAS, 2012; POISEL; MALZER; TJOA, 2013; DYKSTRA; SHERMAN, 2013; LEE; UN, 2012), consiste em coletar dados pertinentes à investigação e armazená-los fora da nuvem; a segunda, empregada em (SANG, 2013) e que constitui um caso específico de (GEORGE; VENTER; THOMAS, 2012), 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 (ALQAHTANY et al., 2015).

3.2.4 Garantir a cadeia de custódia da evidência

Dentre os trabalhos analisados, apenas (SANG, 2013) 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. Uma limitação desse trabalho, entretanto, é que ele não deixa explícitos 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 detalhadamente garantia de custódia. Em geral, os trabalhos apenas mencionam que as evidências devem ser coletadas de forma forensicamente aceitável.

3.3 Resumo

A Tabela 1 mostra um comparativo das soluções estudadas, considerando os aspectos discutidos nesta seção, posicionando as contribuições da proposta apresentada neste trabalho.

Tabela 1: Comparativo de soluções de coleta de informações de memória de máquinas em nuvem para análise forense

	Dizang (esta proposta)	(GEORGE; VENTER; THOMAS, 2012)	(POISEL; MALZER; TJOA, 2013)	(DYKSTRA; SHERMAN, 2013)	(LEE; UN, 2012)	(REICHERT; RICHARDS; YOSHIGOE, 2015)	(SANG, 2013)	(DOLAN-GAVITT et al., 2011)	(DEZFOULI et al., 2012)
Coleta evidência continuamente.	✓	✘	✘	✘	✓	✘	✓	✘	✓
O processo é reprodutível.	✓	✘	✘	✘	✘	✘	✘	✘	✘
Garante cadeia de custódia.	✓	✘	✘	✘	✘	✓	✓	✘	✘
Preserva privacidade e jurisdição.	✓	✓	✓	✓	✓	✓	✓	✓	✓

4 PROPOSTA DE PROJETO: DIZANG

A presente proposta tem como objetivo principal coletar memória de recursos computacionais virtuais em arquitetura volátil 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.

4.1 Identificação da origem

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

Para conseguir correlacionar uma evidência a sua origem volátil, é necessário utilizar outro elemento em que persista a relação fonte-evidência. O presente trabalho

propõe que isto seja feito por meio de cálculo de hash do recurso em nuvem que produziu a evidência. O hash de um recurso em nuvem permite identificar univocamente a fonte de uma evidência. Em arquiteturas que utilizam contêiner por exemplo, é possível identificar se a evidência veio do contêiner do motor de páginas dinâmicas (e.g., Apache), do contêiner da lógica de negócios (e.g., *golang*) ou do contêiner do banco de dados (e.g., *Cassandra*).

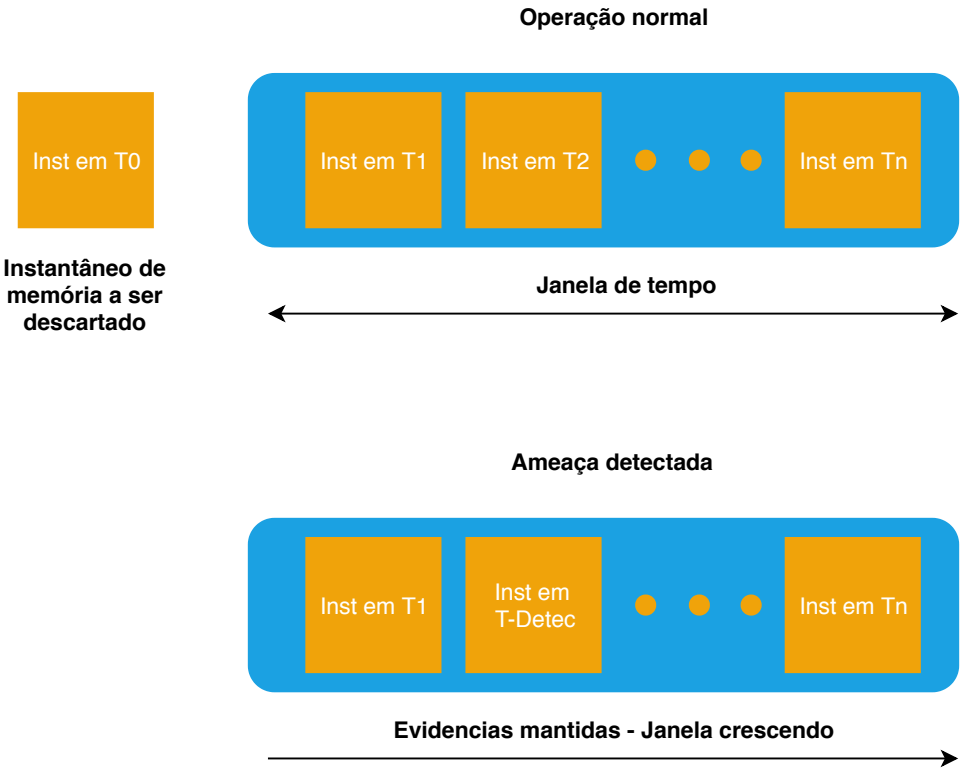
4.2 Descrever o sistema antes e depois do incidente

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 recurso, informações possivelmente importantes para a investigação podem acabar sendo perdidas. Com o objetivo de deixar o processo de cópia da memória mais atômico, a fim de evitar inconsistências na informação coletada (CASE et al., 2014), Dizang propõe que a execução do recurso em nuvem seja temporariamente suspenso para que seja realizada a cópia de sua memória. Essa técnica, que é semelhante àquela adotada em (RAFIQUE; KHAN, 2013) para VMs, produz um instantâneo da memória volátil do recurso; isso permite sua análise em um estado de repouso, ou seja, sem a necessidade de ter o recurso em execução. Ao realizar a coleta em intervalos de tempo adequados, é possível construir um histórico do estado da memória durante a execução no recurso.

A maioria das técnicas forenses mais usadas atualmente são voltadas à obtenção da informação em sua totalidade. Isso comumente é feito via cópia bit a bit ou por meio da obtenção do *hardware* físico (SIMOU et al., 2014) (BEM et al., 2008). 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 (QUICK; CHOO, 2014). Para mitigar essa dificuldade, em Dizang são 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 (CASE et al., 2014). Assim, a solução proposta implementa uma janela de instantâneos de memória cobrindo um intervalo de tempo pré-definido, como ilustrado na Figura 8. 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), Dizang deixa de descartar as coletas mais antigas do *log* de monitoramento. Como resultado, é possível conhecer o sistema antes e depois do ataque e, assim, avaliar sua evolução.

Figura 8: Janela deslizante de coleta de evidência



Fonte: Próprio autor

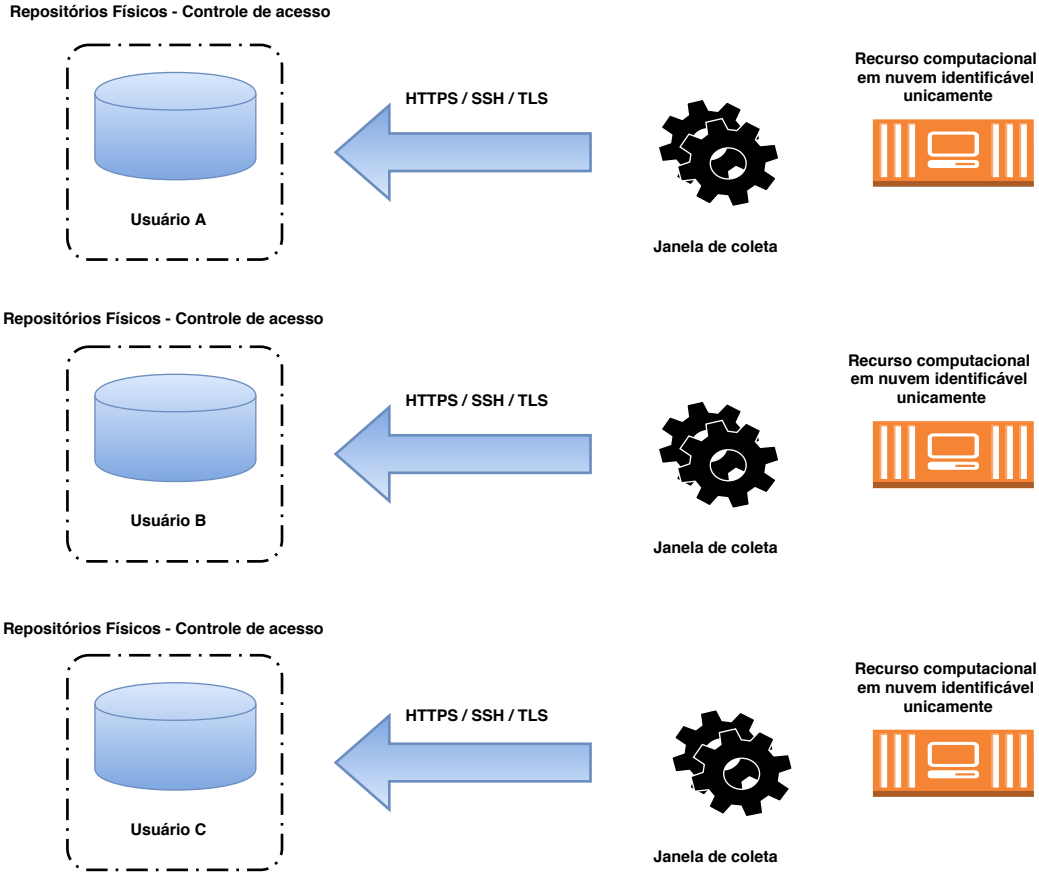
4.3 Garantindo integridade, confidencialidade e protegendo privacidade e jurisdição

Finalmente, para persistir a relação evidência-origem e garantir a sua integridade, Dizang calcula o hash H do par {evidência, identificador da imagem do contêiner} e armazena a tripla {H, identificador do recurso, evidência}. Adicionalmente, a presente proposta evita eventuais problemas com o armazenamento desses dados em países com jurisdições diferentes daquelas que devem ser aplicadas na investigação em questão. Especificamente, as evidências coletadas são armazenadas em um local físico fora da nuvem, após serem transportadas por meio de um canal seguro (e.g., via TLS (*Transport Layer Security* – Camada de Transporte Seguro) (Dierks T, 2008)).

4.4 Implementação

Os mecanismos propostos foram implementados em uma plataforma de testes visando avaliar a eficácia de Dizang em coletar as informações de memória dos contêineres de forma reprodutível, sem violar jurisdições ou a privacidade de usuários e a capacidade de detectar injeção de código usando as evidências coletadas. A solução, ilustrada na Figura 9, consistiu na criação de uma instancia *t2.micro* na zona Ohio da AWS com 3.3Mhz, 1Gb de RAM e sistema operacional de 64 bits. Nesta instância AWS foi instalado o Docker Engine 1.10 e a API Docker 1.21, com os quais foram criados 3 contêineres executando o nginx 1.0 em diferentes portas. Foi desenvolvida uma aplicação Java que, executada no sistema operacional hospedeiro, descobre o identificador de processo associado a cada contêiner, copia o conteúdo do *descritor de alocação de memória não uniforme* (**/proc/pid/numa_maps**), o qual contém a alocação das páginas de memória, os nós que estão associados a essas páginas, o que está alocado e suas respectivas políticas de acesso (Unix Man Pages, e). A cópia e gravação do arquivo é tal que, a cada intervalo de tempo t, a aplicação (1) pausa o contêiner em

Figura 9: Arquitetura geral da solução Dizang



Fonte: Próprio autor

questão, (2) copia a diretório **numa_maps**, (3) concatena os dados obtidos com o identificador da imagem e do contêiner, (4) calcula o H do conjunto e (5) salva o resultado em um arquivo cujo nome é o identificador da imagem e do contêiner e a extensão é **.mem**. O transporte seguro da evidência para um armazenamento físico fora da AWS foi implementado usando uma instância *t2.micro* na zona Ohio da AWS onde foi instalado um servidor *OpenVPN*. Como uma forma básica de controle de acesso, a instância EC2 que contém as evidências foi configurada para aceitar conexões apenas de máquinas nesta VPN. Uma máquina física fora da AWS, usou o cliente do *OpenVPN* para estabelecer uma conexão VPN com a instância que contém as evidências e as transportou para o disco da máquina física. Após a conclusão do processo de transporte, a máquina física verifica se existem arquivos **.mem** em disco mais antigos que um certo

intervalo de tempo *t*, descartando-os.

4.5 Resultados experimentais

Para avaliar a efetividade de Dizang na coleta de evidências e identificação de injeção de código, dois experimentos foram realizados usando o ambiente implementado (descrito na Seção 4.4).

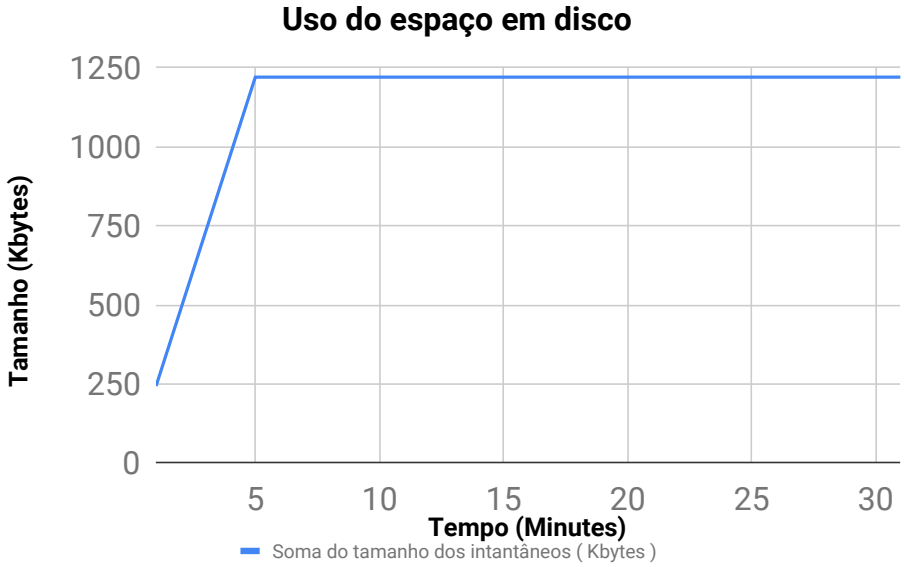
4.5.1 Análise do desempenho

No primeiro experimento, o sistema foi configurado para realizar coletas de memória em intervalos de 1 minuto, salvá-las em armazenamento externo à nuvem e apagar amostras coletadas há mais de 5 minutos. O sistema foi então executado por 30 minutos, tempo durante o qual foram coletadas como métricas (1) o uso de espaço em disco utilizado pelos instantâneos de memória salvos, (2) o tempo de pausa no contêiner necessário para a cópia delas e (3) o tempo de transporte das evidências para o armazenamento externo a nuvem.

A evolução do espaço em disco ocupado pelos instantâneos de memória, acompanhado através da execução do comando `du -sh *.mem` do *Unix* no disco de armazenamento externo, é mostrada no gráfico da Figura 10. Neste experimento os instantâneos de memória tem 244kb de tamanho. 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. Assim, a solução mantém sob controle o espaço em disco ocupado pelas amostras coletadas. Ao mesmo tempo, instantâneos de memória salvos pela solução depois que os contêineres são removidos continuam no disco da máquina, podendo ser associados a sua origem (i.e., contêiner e imagem), conforme esperado para uma análise forense. Essa capacidade se mantém após a detecção de uma ameaça, pois

nesse caso coletas mais antigas deixam de ser apagadas. Logo, é possível descrever o estado do sistema antes e depois do incidente (CASE et al., 2014), permitindo-se, por exemplo, que ataques de injeção de código em memória sejam analisados.

Figura 10: Evolução do uso do espaço em disco com o Dizang

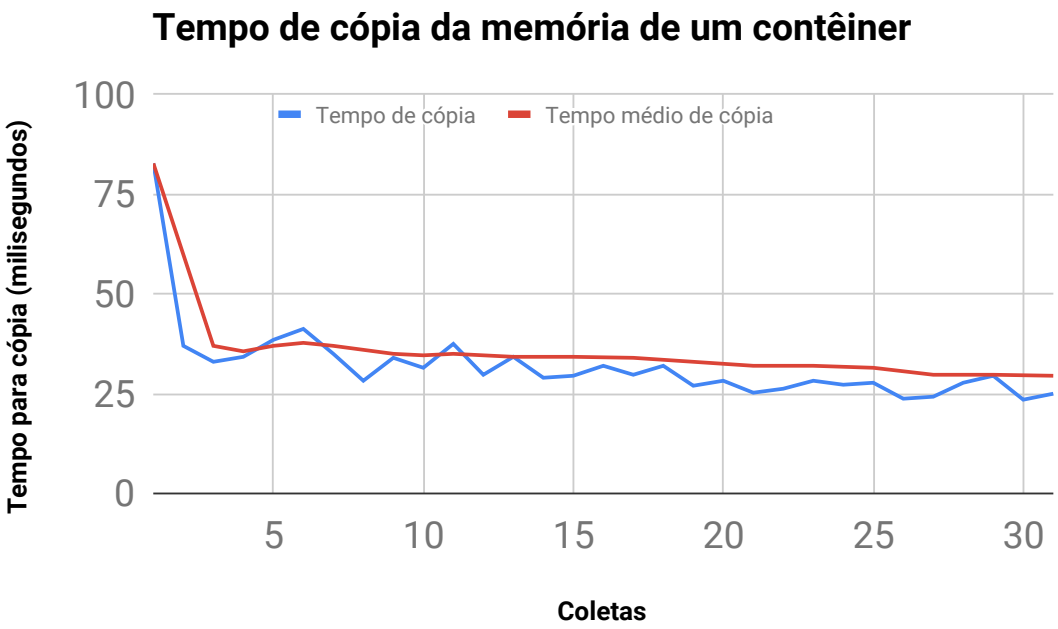


Fonte: Próprio autor

Uma potencial limitação da solução proposta é que a pausa de um contêiner para coleta de dados poder, em princípio, causar perdas no desempenho da aplicação sendo executada. Para avaliar esse impacto, durante o experimento foram medidos os tempos de cópia da memória do contêiner. Os resultados são mostrados no gráfico da Figura 11. É possível notar que, após a inicialização da aplicação, o tempo para realizar a cópia é bastante reduzido, variando entre 20 e 40 milissegundos. Em especial, para contêineres executando um motor de páginas web dinâmicas, como é o caso do experimento em questão, essa latência deve ser pouco perceptível por usuários finais. Para os casos em que a interrupção da execução do recurso computacional mesmo por breves momentos cause problemas de disponibilidade, é possível realizar o procedimento de coleta em instantes de tempo separados. Assim, ao invés de suspender a execução de todos os recursos computacionais para realização da coleta simultaneamente, o procedimento interrompe-as sequencialmente. Desta forma, a latência demonstrada pode

ser considerado o pior caso neste experimento.

Figura 11: Tempo de cópia da memória de um contêiner



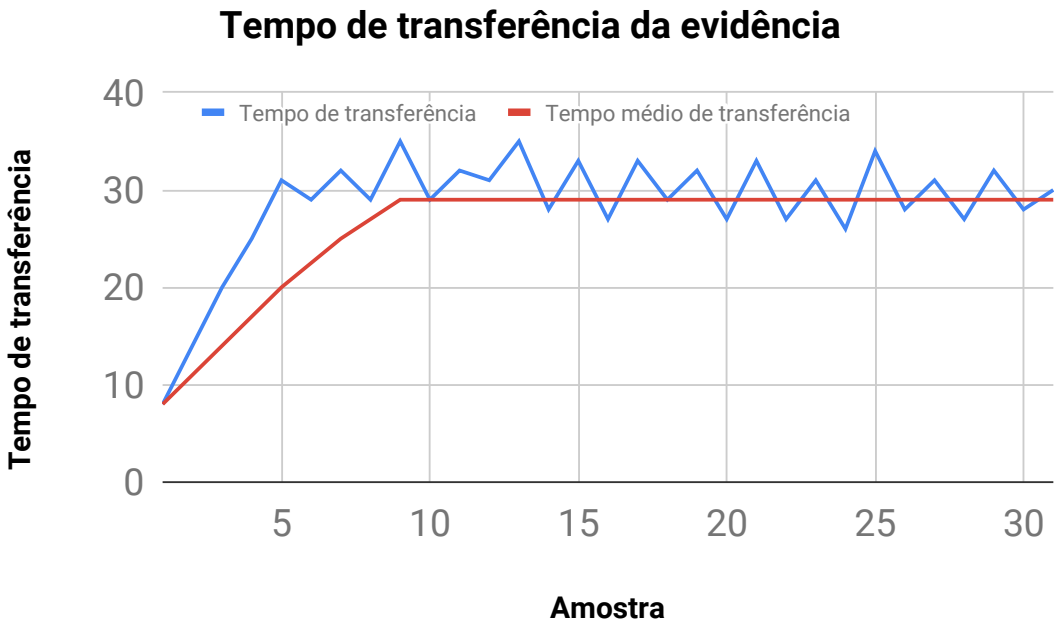
Fonte: Próprio autor

Outra preocupação é o tempo de transporte das evidências para o armazenamento fora da nuvem. Caso o transporte da evidência leve mais tempo que a geração do próximo instantâneo, um backlog de transporte se formará levando a perdas nas evidências que estejam pendentes para transporte. Para avaliar esse impacto, durante o experimento foram medidos os tempos de transporte das evidências para o armazenamento fora da nuvem. Os resultados são mostrados no gráfico da Figura 12. É possível notar que o tempo de transporte estabiliza após atingido o tamanho da janela. O tempo de transporte da evidência fica, em média próximo dos 30 segundos.

Tanto a topologia quando a arquitetura do transporte da evidência e a arquitetura do que se deseja extrair a evidência são fatores que contribuem tanto positiva quando negativamente no tempo de transporte. Neste experimento o gerador de evidências, um motor de páginas dinâmicas, está na América do Norte enquanto que a máquina

física para onde as evidências foram transportadas e que é responsável pelo transporte da evidência está na América do Sul.

Figura 12: Tempo de transporte da evidência



Fonte: Próprio autor

4.5.2 Identificação de injeção de código malicioso

Um segundo experimento teve como objetivo determinar se é possível, através da análise das evidências coletadas, identificar injeção de código malicioso na memória do contêiner. Para este fim uma biblioteca **libexample.so** simulando um código malicioso foi injetado em um dos contêineres. Após cinco minutos de Dizang realizando coletas, uma biblioteca foi injetada na memória de um dos contêineres. Após a injeção permitiu-se que a solução continuasse coletando por mais 5 minutos. Além da coleta do conteúdo do diretório **/proc/pid/numa_maps**, realizou-se também uma cópia crua da memória do processo do contêiner utilizando o utilitário *nsenter* via comando descrito na Figura 13.

Figura 13: Comando para cópia crua da memória do processo do contêiner

```
sudo ./nsenter --target <pid> --mount --uts --ipc --net --pid

grep rw-p /proc/<pid>/maps | sed -n 's/^\([0-9a-f]*\)-\([0-9a-f]*\) .*$/\1 \2/p' |
while read start stop; do gdb --batch --pid <pid> -ex "x/4096wx 0x$start";
done > <pid>.dump
```

Fonte: Próprio autor

De posse das coletas do diretório **/proc/pid/numa_maps** comparou-se dois momentos distintos na vida do contêiner, antes e depois da injeção da biblioteca. Observando as Figuras 14 e 15 é possível notar que no instantâneo após a injeção aparece a biblioteca **libexample.so** simulando o código malicioso entre os endereços **7f85631b8000** e **7f85633b9000**. Logo, é possível identificar a injeção de um código malicioso via evidência coletada por Dizang do diretório **/proc/pid/numa_maps**, permitindo-se por exemplo que ataques de injeção de código sejam identificados. A Figura 16 mostra o conteúdo da parte legível da memória no endereço **0x7f85633b9000** onde a biblioteca **libexample.so** simulando um código malicioso está alocada.

Figura 14: Parte do arquivo **/proc/pid/numa_maps** ANTES da injeção

```
00400000 default file=/home/someuser/target mapped=1 N0=1
00600000 default file=/home/someuser/target anon=1 dirty=1 N0=1
00601000 default file=/home/someuser/target anon=1 dirty=1 N0=1
7fee0b0ed000 default file=/lib/x86_64-linux-gnu/libc-2.19.so mapp...
7fee0b2ab000 default file=/lib/x86_64-linux-gnu/libc-2.19.so
7fee0b4ab000 default file=/lib/x86_64-linux-gnu/libc-2.19.so anon...
7fee0b4af000 default file=/lib/x86_64-linux-gnu/libc-2.19.so anon...
7fee0b4b1000 default anon=3 dirty=3 N0=3
7fee0b4b6000 default file=/lib/x86_64-linux-gnu/ld-2.19.so mapp...
7fee0b6d1000 default anon=3 dirty=3 N0=3
7fee0b6d6000 default anon=2 dirty=2 N0=2
7fee0b6d8000 default file=/lib/x86_64-linux-gnu/ld-2.19.so anon...
7fee0b6d9000 default file=/lib/x86_64-linux-gnu/ld-2.19.so anon...
7fee0b6da000 default anon=1 dirty=1 N0=1
7ffc91a60000 default stack anon=2 dirty=2 N0=2
7ffc91b15000 default
7ffc91b17000 default
```

Fonte: Próprio autor

Figura 15: Parte do arquivo **/proc/pid/numa_maps** APÓS a injeção

```
00400000 default file=/home/someuser/target map...
00600000 default file=/home/someuser/target ano...
00601000 default file=/home/someuser/target ano...
7f8562def000 default file=/lib/x86_64-linux-gnu/lib...
7f8562fad000 default file=/lib/x86_64-linux-gnu/lib...
7f85631ad000 default file=/lib/x86_64-linux-gnu/lib...
7f85631b1000 default file=/lib/x86_64-linux-gnu/lib....
7f85631b3000 default anon=3 dirty=3 N0=3
7f85631b8000 default file=/home/someuser/libexample.so ...
7f85631b9000 default file=/home/someuser/libexample.so
7f85633b8000 default file=/home/someuser/libexample.so ...
7f85633b9000 default file=/home/someuser/libexample.so ...
7f85633ba000 default file=/lib/x86_64-linux-gnu/ld-2.19.so...
7f85635d4000 default anon=3 dirty=3 N0=3
7f85635d9000 default anon=3 dirty=3 N0=3
7f85635dc000 default file=/lib/x86_64-linux-gnu/ld-2.19.so...
7f85635dd000 default file=/lib/x86_64-linux-gnu/ld-2.19.so...
7f85635de000 default anon=1 dirty=1 N0=1
7fff158b7000 default stack anon=4 dirty=4 N0=4
7fff159b2000 default
7fff159b4000 default
```

Fonte: Próprio autor

Figura 16: Conteúdo da memória de **libexample.so** no formato [endereço]: [conteúdo]

```
0x7f85633b9000: 0x00200e08 0x00000000 0x635db000 0x00007f85
0x7f85633b9010: 0x633d0570 0x00007f85 0x62e11760 0x00007f85
0x7f85633b9020: 0x62ede3b0 0x00007f85 0x631b8616 0x00007f85
0x7f85633b9030: 0x631b8626 0x00007f85 0x633b9038 0x00007f85
0x7f85633b9040: 0x00000000 0x00000000 0x75746e75 0x382e3420
0x7f85633b9050: 0x322d342e 0x6e756275 0x7e317574 0x302e3431
0x7f85633b9060: 0x29342e34 0x382e3420 0x0000342e 0x6d79732e
0x7f85633b9070: 0x00626174 0x7274732e 0x00626174 0x7368732e
0x7f85633b9080: 0x61747274 0x6e2e0062 0x2e65746f 0x2e756e67
0x7f85633b9090: 0x6c697562 0x64692d64 0x6e672e00 0x61682e75
0x7f85633b90a0: 0x2e006873 0x736e7964 0x2e006d79 0x736e7964
0x7f85633b90b0: 0x2e007274 0x2e756e67 0x73726576 0x006e6f69
0x7f85633b90c0: 0x756e672e 0x7265762e 0x6e6f6973 0x2e00725f
0x7f85633b90d0: 0x616c6572 0x6e79642e 0x65722e00 0x702e616c
0x7f85633b90e0: 0x2e00746c 0x74696e69 0x65742e00 0x2e007478
0x7f85633b90f0: 0x696e6966 0x6f722e00 0x61746164 0x68652e00
0x7f85633b9100: 0x6172665f 0x685f656d 0x2e007264 0x665f6865
0x7f85633b9110: 0x656d6172 0x6e692e00 0x615f7469 0x79617272
0x7f85633b9120: 0x69662e00 0x615f696e 0x79617272 0x636a2e00
0x7f85633b9130: 0x642e0072 0x6d616e79 0x2e006369 0x00746f67
0x7f85633b9140: 0x746f672e 0x746c702e 0x61642e00 0x2e006174
```

Fonte: Próprio autor

4.6 Limitações

A proposta descrita pede que o recurso em nuvem seja identificável de forma única a fim de realizar a associação entre evidência e sua origem. Durante o curso deste projeto essa identificação única só foi possível através do hash da imagem do contêiner. Este foi o único recurso que, submetido ao processo de construção a partir da mesma receita resultou no mesmo hash da imagem. Assim, a implementação para verificação da solução proposta consegue apenas coletar informações de memória no espaço do usuário (*user space*), ela não consegue acessar o espaço de kernel (*kernel space*). A implementação de Dizang neste documento em princípio não consegue investigar códigos malicioso que se baseiam em informações do *kernel space*. Isso inclui, por exemplo, a comparação de informações do PEB (*Process Environment Block* – Bloco para o Ambiente dos Processos), que ficam no *user space*, com informações do VAD (*Virtual Address Descriptor* – Descritor de Endereços de Memória Virtual), que fica no *kernel space*.

Outra limitação da solução proposta é a necessidade da mesma estar instalada no sistema sob investigação a priori para que os resultados descritos neste documento sejam alcançados. Como mencionado em (CASE et al., 2014), "para uma análise eficiente de um incidente em memória, são necessárias cópias da mesma **antes e depois** do incidente."

5 CONCLUSÕES E RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Neste capítulo são apresentadas as conclusões do presente trabalho e as recomendações para a continuidade dos trabalhos neste campo de estudo.

5.1 Conclusões

Ameaças digitais que atuam diretamente na memória de sistema não costumam deixar rastros em disco após terem os recursos correspondentes desalocados, dificultando análises forenses posteriores. Esse problema é especialmente notável em sistemas de computação em nuvem, nos quais a alocação e desalocação de recursos virtualizados (e.g., VMs e contêineres) é frequente. Essa característica, aliada a aspectos como multi-inquilinato e multi-jurisdição de nuvens computacionais, dificulta a coleta de evidências para a investigação de incidentes.

Nesse cenário, a proposta apresentada visa relacionar o instantâneo de memória a sua origem, utilizando o *hash* calculado do recurso computacional em nuvem como identificador da origem da evidência armazenada. Para evitar uso excessivo de memória, a quantidade de dados armazenados usa uma janela de armazenamento, o que permite descrever a memória antes e depois de um ataque (e.g., de injeção de memória). Transportando de forma segura e armazenando a evidência em local conhecido fora da nuvem, evitam-se os problemas relacionados a multi-jurisdição e multi-inquilinato das nuvens computacionais. A comparação de instantâneos de memória coletados em

diferentes instantes de tempo permite a identificação de injeção de código assim como extrair o conteúdo da parte legível do endereço de memória correspondente.

Combinada com uma ferramenta para identificação de ameaças, essas características de Dizang o transformam em uma solução poderosa para prover evidências e, assim, viabilizar análises forenses na nuvem.

5.2 Trabalhos Futuros

Como mencionado em 4.6 a solução proposta é capaz de gerar evidências de memória apenas do espaço do usuário (*user space*). Alguns códigos malicioso injetados em memória são capazes de manipular o retorno de funções do kernel. Recomenda-se para trabalhos futuros a incorporação à Dizang uma forma de realizar a extração do conteúdo de memória do espaço do kernel (*kernel space*).

REFERÊNCIAS

Ab Rahman, N. H.; CHOO, K. K. R. A survey of information security incident handling in the cloud. *Computers and Security*, Elsevier Ltd, vol. 49, p. 45–69, 2015. ISSN 01674048. Available from Internet: <http://dx.doi.org/10.1016/j.cose.2014.11.006>.

ALQAHTANY, S. et al. Cloud forensics: A review of challenges, solutions and open problems. In *Int. Conference on Cloud Computing (ICCC)*. Plymouth, UK: IEEE, 2015. p. 1–9. Available from Internet: <https://ieeexplore.ieee.org/document/7149635>.

BEM, D. et al. Computer forensics - past , present and future. *Journal of Information Science and Technology*, vol. 5, no. 3, p. 43–59, 2008. Available from Internet: <http://www.cis.gsu.edu/rbaskerville/cis8630/Bernetal2008.pdf>.

BERNSTEIN, D. Containers and cloud: From LXC to Docker to Kubernetes. *IEEE Cloud Computing*, vol. 1, no. 3, p. 81–84, Sept 2014. ISSN 2325-6095. Available from Internet: <https://ieeexplore.ieee.org/document/7036275>.

CASE, A. et al. *The Art of Memory Forensics: Detecting malware and threats in Windows, Linux and Mac memory*. Hoboken, NJ: Wiley, 2014.

Chef. *Chef*. <https://www.chef.io>. Acessado em: 16-04-2018.

DEZFOULI, F. et al. Volatile memory acquisition using backup for forensic investigation. In *Int. Conf. on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*. Plymouth, UK: IEEE, 2012. p. 186–189. ISBN 978-1-4673-6618-2. Available from Internet: <https://ieeexplore.ieee.org/document/6246108>.

Dierks T, R. E. *The Transport Layer Security (TLS) Protocol*. Fremont, CA: IETF, 2008. IETF: <https://tools.ietf.org/html/rfc5246>.

DOLAN-GAVITT, B. et al. Virtuoso: Narrowing the semantic gap in virtual machine introspection. In *IEEE Symposium on Security and Privacy*. Plymouth, UK: IEEE, 2011. p. 297–312. ISSN 1081-6011. Available from Internet: <https://ieeexplore.ieee.org/document/5958036>.

DYKSTRA, J.; SHERMAN, A. Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques. *Digital Investigation*, Elsevier Ltd, vol. 9, p. S90–S98, 2012. ISSN 17422876. (Proc. of the 12th Annual DFRWS Conference). Available from Internet: <http://dx.doi.org/10.1016/j.diin.2012.05.001>.

DYKSTRA, J.; SHERMAN, A. T. Design and implementation of FROST: Digital forensic tools for the OpenStack cloud computing platform. *Digital Investigation*, Elsevier Ltd, vol. 10, p. S87–S95, 2013. ISSN 17422876. (Proc. of 13th Annual DFRWS Conference). Available from Internet: <http://dx.doi.org/10.1016/j.diin.2013.06.010>.

FEWER, B. S. Reflective DLL Injection. no. October, 2008. Available from Internet: <https://www.dc414.org/wp-content/uploads/2011/01/242.pdf>.

GEORGE, S.; VENTER, H.; THOMAS, F. Digital Forensic Framework for a Cloud Environment. In *IST Africa Conference Report*. Tanzania: IIMC, 2012. p. 1–8. ISBN 978-1-905824-34-2. Available from Internet: <https://pdfs.semanticscholar.org/104f/cc8b4b9ddc4926d76bb85ce564adcaaf70b4.pdf>.

GOOGLE. *GRR Rapid Response*. 2013. <https://grr-doc.readthedocs.io/en/v3.2.1/what-is-grr.html>. Acessado em: 27-07-2019.

GRISPOS, G.; STORER, T.; GLISSON, W. Calm before the storm: the challenges of cloud computing in digital forensics. *International Journal of Digital Crime and Forensics*, vol. 4, no. 2, p. 28–48, 2012. ISSN 1466640073. Available from Internet: <www.igi-global.com/article/calm-before-storm/68408>.

HBase. *HBaset*. <https://hbase.apache.org/>. Acessado em: 16-06-2018.

KEYUN, R. et al. Advances in Digital Forensics VII. In GILBERT, P.; SUJEET, S. (Ed.). *7th IFIP WG 11.9 International Conference on Digital Forensics*. 7. ed. Orlando: [s.n.], 2011. vol. 1, chap. 3. ISBN 9788578110796. Available from Internet: <https://www.springer.com/gp/book/9783642242113>.

LEE, J.; UN, S. Digital Forensics as a Service: A case study of forensic indexed search. p. 499–503, 2012. Available from Internet: <https://ieeexplore.ieee.org/document/6387185>.

LINUXCONTAINERS.ORG. *Linux Containers (LXC)*. 2015. <https://linuxcontainers.org/lxc/introduction/>. Acessado em: 20/05/2017.

LUIS, P.; SANCHEZ, P.; GIOVA, G. Digital Chain of Custody Quality Assessment. *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 16, no. 4, p. 41–48, 2016. ISSN 17387906. Available from Internet: <https://www.researchgate.net/publication/303250922_Digital_Chain_of_Custody_Quality_Assessment>.

MELL, P.; GRANCE, T. *The NIST definition of cloud computing*. 2011. 7 p. NIST SP 800-145. Available from Internet: <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>.

MILLER, M.; TURKULAINEN, J. *Remote Library Injection*. 2004. Tech. Report: <http://www.hick.org/code/skape/papers/remote-library-injection.pdf>.

MORSY, A. M.; GRUNDY, J.; MULLER, I. An Analysis of the Cloud Computing Security Problem. In *APSEC Cloud Workshop*. Sydney, Australia: Cornell University, 2010. Available from Internet: <https://arxiv.org/abs/1609.01107>.

POISEL, R.; MALZER, E.; TJOA, S. Evidence and cloud computing: The virtual machine introspection approach. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 4, no. 1, p. 135–152, 2013. ISSN 20935374 (ISSN). Available from Internet: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.469.937>.

Puppet. *Puppet*. <https://puppet.com/>. Acessado em: 16-04-2018.

QUICK, D.; CHOO, K. K. R. Impacts of increasing volume of digital forensic data: A survey and future research challenges. *Digital Investigation*, Elsevier Ltd, vol. 11, no. 4, p. 273–294, 2014. ISSN 17422876. Available from Internet: <http://dx.doi.org/10.1016/j.diin.2014.09.002>.

RAFIQUE, M.; KHAN, M. N. A. Exploring Static and Live Digital Forensics: Methods, Practices and Tools. *International Journal of Scientific & Engineering Research*, vol. 4, no. 10, p. 1048–1056, 2013. Available from Internet: <www.ijser.org/researchpaper/Exploring-Static-and-Live-Digital-Forensic-Methods-Practices-and-Tools.pdf>.

RAHMAN, N. H. A.; CAHYANI, N. D. W.; CHOO, K.-K. R. Cloud incident handling and forensic-by-design: cloud storage as a case study. *Concurrency Computation Practice and Experience*, p. 685–701, 2016. Available from Internet: <https://www.researchgate.net/publication/303373633_Cloud_incident_handling_and_forensic-by-design_cloud_storage_as_a_case_study_Cloud_Incident_Handling_and_Forensic-by-Design>.

RAHMAN, S.; KHAN, M. N. A. Review of live forensic analysis techniques. *International Journal of Hybrid Information Technology*, vol. 8, no. 2, p. 379–388, 2015. Available from Internet: <https://www.researchgate.net/publication/283740637_Review_of_Live_Forensic_Analysis_Techniques>.

RAMOS, M. G. *Do Valor Probatório do Arquivo Digital*. PhD Thesis (PhD) — Universidade de Brasilia, 2011. Available from Internet: <http://bdm.unb.br/bitstream/10483/1843/1/2011_MonicaGomesRamos.pdf>.

REICHERT, Z.; RICHARDS, K.; YOSHIGOE, K. Automated forensic data acquisition in the cloud. *IEEE Int. Conf. on Mobile Ad Hoc and Sensor Systems*, p. 725–730, 2015. Available from Internet: <https://ieeexplore.ieee.org/document/7035772>.

Right Scale. *State of the Cloud Report*. 2018. Available from Internet: <www.suse.com/media/report/rightscale_2018_state_of_the_cloud_report.pdf>.

SANG, T. A log-based approach to make digital forensics easier on cloud computing. *2013 3rd International Conference on Intelligent System Design and Engineering Applications, ISDEA 2013*, p. 91–94, 2013. Available from Internet: <https://ieeexplore.ieee.org/document/6454779>.

SIMOU, S. et al. Cloud forensics: Identifying the major issues and challenges. In *Advanced Information Systems Engineering (CAiSE 2014)*. Cham, CH:

Springer International Publishing Switzerland 2014, 2014. vol. 8484, p. 271–284. ISBN 9783319078809. ISSN 16113349. Available from Internet: <https://www.researchgate.net/publication/289531775_Cloud_Forensics_Identifying_the_Major_Issues_and_Challenges>.

STACK, O. *Open Stack Framework*. 2018. <https://www.openstack.org>. Acessado em: 20-05-2018.

Unix Man Pages. *cgroups - Control Groups*. <http://man7.org/linux/man-pages/man7/cgroups.7.html>. Acessado em: 15-09-2017.

____. *chroot - Change Root coomand*. <http://man7.org/linux/man-pages/man1/chroot.1.html>. Acessado em: 24-09-2017.

____. *dd - convert and copy a file*. <http://man7.org/linux/man-pages/man1/dd.1.html>. Acessado em: 01-12-2017.

____. *Namespacing*. <http://man7.org/linux/man-pages/man7/namespaces.7.html>. Acessado em: 15-09-2017.

____. *Numa Maps - Non Uniform Memory Architecture*. <http://man7.org/linux/man-pages/man7/numa.7.html>. Acessado em: 24-06-2017.

Vagrant. *Vagrant*. <https://www.vagrantup.com/>. Acessado em: 16-04-2018.

VöMEL, S.; STüTTGEN, J. An evaluation platform for forensic memory acquisition software. In . Amsterdam, The Netherlands, The Netherlands: Elsevier Science Publishers B. V., 2013. vol. 10, p. S30–S40. ISSN 1742-2876. Elsevier Science Publishers. Available from Internet: <http://dx.doi.org/10.1016/j.diin.2013.06.004>.

WIKIPEDIA. *MapReduce*. 2018. <https://en.wikipedia.org/wiki/MapReduce>. Acessado em: 29 janeiro 2018.