Influência de fatores humanos na análise forense de máquinas em núvem

Hamilton J. S. Fonte II

[1]Escola Politécnica – Universidade de São Paulo (USP)
Av. Prof. Luciano Gualberto 380 - Butantã – 05.508-010 – São Paulo – SP – Brasil

Resumo. *Arquiteturas em nuvem são cada vez mais comuns, e também o número de problemas de segurança envolvendo essa tecnologia. Infelizmente, devido à natureza volátil de recursos na nuvem, a coleta de evidências para análise forense nesse ambiente tem esbarrado em desafios práticos e legais. Específico a disciplina de fatores humanos, este trabalho analisa o impacto do crescente volume de informações que os analistas forenses precisam analisar como um dos desafios trazidos pela nuvem à forense computacional.*

1. Introdução

Técnicas de virtualização, replicação de serviços e compartilhamento de recursos entre múltiplos usuários (multi-inquilinato) dão a nuvens computacionais alta escalabilidade [Morsy et al. 2010]. Embora interessante do ponto de vista de eficiência e custos, do ponto de vista forense tais mecanismos trouxeram uma série de novos desafios para a forense computacional. Por exemplo, com a facildade de alocação de recursos, o volume de dados que uma instância em nuvem é capaz de armazenar tornou a transferência desses dados para laboratório muito custosa, sem falar na análise de todos esses dados. Essa dificuldade é ainda agravada por aspectos como multi-inquilinato e multi-jurisdição típicas de soluções em nuvem [Gilbert and Sujeet 2008]. 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 [Dykstra and Sherman 2012]. 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 [Rahman and Khan 2015].

2. 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. Há três modelos principais de comercialização de uso da nuvem [Mell and Grance 2011]: *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 VMs, vem sendo crescentemente feita também na forma de contêineres. De fato, segundo estudo realizado em 2016 com 235 empresas que têm desenvolvimento de software como sua atividade fim ou como suporte à atividade fim [DevOps and ClusterHQ 2016], 76% dos respondentes utilizam contêineres para melhorar a eficiência do processo de desenvolvimento e em suas arquiteturas de micro-serviços em nuvem. Diferentemente de VMs, que envolvem a criação de um *hardware* virtual e também de um SO acima do sistema nativo, 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 Contêineres Linux (LXC) [Linuxcontainers.org 2015], que aproveitam-se de funcionalidades como cgroups e namespacing do kernel do Linux para auxiliar no gerenciamento e isolamento de recursos virtuais.

3. Trabalhos relacionados

Focando nos fatores humanos da forense computacional, analisamos alguns dos trabalhos existentes.

3.1. Acessar e coletar as informações de memória na 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 et al. 2015, Poisel et al. 2013, Dykstra and Sherman 2013, George et al. 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 técnicas de análise executam comparações entre o estado do sistema antes e depois da intrusão. 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 et al. 2013, Dykstra and Sherman 2013, Sang 2013], que adotam a estratégia de isolar e parar a VM 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.

3.2. 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 [Reichert et al. 2015, George et al. 2012, Poisel et al. 2013,

Dykstra and Sherman 2013], 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 et al. 2012], depende da cooperação do provedor de serviços de nuvem para conseguir as informações necessárias à investigação.

4. Solução proposta: Dizang

A presente proposta, focada na disciplina de fatores humanos tem como objetivo principal manter sob controle a quantidade de dados que os analistas forenses tem que trabalhar sem violar a jurisdição ou privacidade dos não envolvidos na investigação. A solução aqui apresentada, denominada Dizang , é descrita em detalhes a seguir.

4.1. Descrição

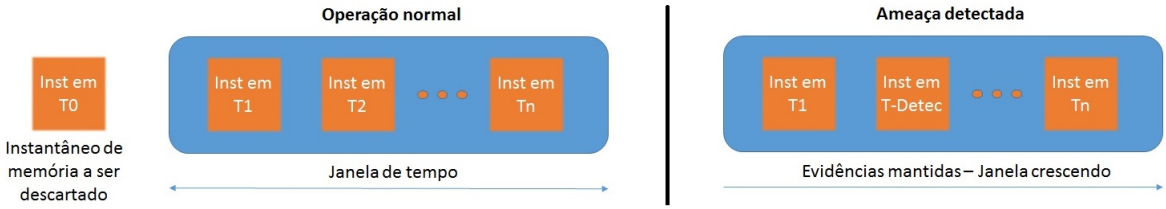
Em sistemas computacionais executados sobre uma infraestrutura física (i.e., não virtualizada), pode-se fazer uma associação direta entre um recurso qualquer, como uma informação da memória, imagem de disco ou pacotes trafegando na rede, e sua origem correspondente. Já em sistemas construídos sobre uma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

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, Dizang interrompe 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 [Rafique and Khan 2013] para VMs, 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 [Simou et al. 2014] [Bem et al. 2008]. Embora tais técnicas possam parecer interessantes à primeira vista, elas muitas vezes acabam sendo responsáveis pelo crescente volume de informações que os investigadores precisam analisar [Quick and 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 Fig. 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), Dizang deixa 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.

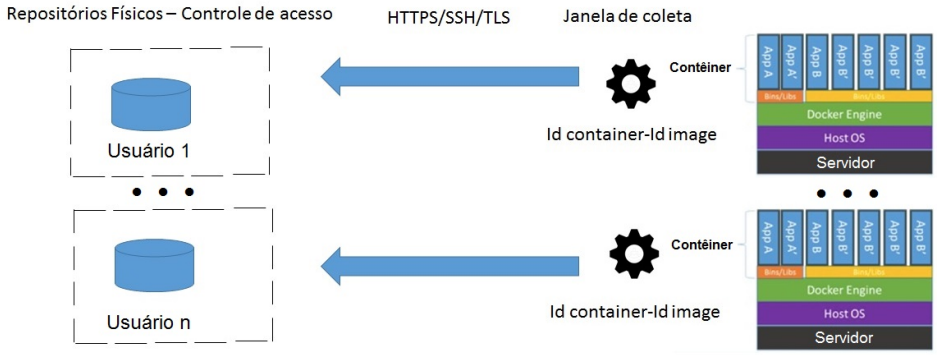
Figura 1. Janela deslizante de coleta de evidência



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 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., via *Transport Layer Security* – TLS [Dierks T 2008]).

4.2. Implementação

Figura 2. Arquitetura geral da solução Dizang



Os métodos 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. A solução, ilustrada na Fig. 2, consistiu na criação de 1 VM usando o Oracle Virtual Box 5.0em um notebook Intel i5 de 2.30Mhz e 4Gb de RAM com sistema operacional de 64 bits. Essas VMs possuem 2 Gb de memória RAM e emulam apenas 1 processador, e em cada uma delas foi instalado o Docker Engine 1.10e a API Docker 1.21, com os quais foram criados 3 contêineres executando o nginx 1.0em diferentes portas. Usando uma aplicação Java que descobre o identificador de processo associado a cada contêiner, pode-se copiar 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]. A cópia e gravação do arquivo é tal que, 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 que um certo intervalo de tempo "t", descartando-os.

5. Considerações Finais

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.

A estratégia de coleta usa uma janela de armazenamento, o que permite descrever a memória antes e depois de um ataque (e.g., de injeção de memória) e mantem sob controle a quantidade de dados coletados para análise. 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.

Referências

Bem, D., Feld, F., Huebner, E., and Bem, O. (2008). Computer forensics - past , present and future. *Journal of Information Science and Technology*, 5(3):43–59.

Case, A., Ligh, M., Jamie, L., and Walters, A. (2014). *The Art of Memory Forensics: Detecting malware and threats in Windows, Linux and Mac memory*. Wiley.

DevOps and ClusterHQ (2016). Container market adoption survey 2016. `https://clusterhq.com/assets/pdfs/state-of-container-usage-june-2016.pdf`.

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

Dykstra, J. and Sherman, A. (2012). Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques. *Digital Investigation*, 9:S90–S98. (Proc. of the 12th Annual DFRWS Conference).

Dykstra, J. and Sherman, A. T. (2013). Design and implementation of FROST: Digital forensic tools for the OpenStack cloud computing platform. *Digital Investigation*, 10:S87–S95. (Proc. of 13th Annual DFRWS Conference).

George, S., Venter, H., and Thomas, F. (2012). Digital Forensic Framework for a Cloud Environment. In *IST Africa*, pages 1–8, Tanzania. IIMC.

Gilbert, P. and Sujeet, S. (2008). *Advances in Digital Forensics IV*, volume 1. Springer-US, Orlando, 1 edition.

Linuxcontainers.org (2015). Linux Containers (LXC).

Mell, P. and Grance, T. (2011). The NIST definition of cloud computing. NIST SP 800-145. `csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf`.

Morsy, A. M., Grundy, J., and Muller, I. (2010). An Analysis of the Cloud Computing Security Problem. In *APSEC Cloud Workshop*, Sydney, Australia.

Poisel, R., Malzer, E., and Tjoa, S. (2013). Evidence and cloud computing: The virtual machine introspection approach. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 4(1):135–152.

Quick, D. and Choo, K. K. R. (2014). Impacts of increasing volume of digital forensic data: A survey and future research challenges. *Digital Investigation*, 11(4):273–294.

Rafique, M. and Khan, M. N. A. (2013). Exploring Static and Live Digital Forensics: Methods, Practices and Tools. *IJSER*, 4(10):1048–1056.

Rahman, S. and Khan, M. N. A. (2015). Review of live forensic analysis techniques. *International Journal of Hybrid Information Technology*, 8(2):379–388.

Reichert, Z., Richards, K., and Yoshigoe, K. (2015). Automated forensic data acquisition in the cloud. *IEEE Int. Conf. on Mobile Ad Hoc and Sensor Systems*, pages 725–730.

Sang, T. (2013). A log-based approach to make digital forensics easier on cloud computing. *Intelligent System Design and Engineering Applications (ISDEA)*, pages 91–94.

Simou, S., Kalloniatis, C., Kavakli, E., and Gritzalis, S. (2014). Cloud forensics: Identifying the major issues and challenges. In *Advanced Information Systems Engineering (CAiSE 2014)*, volume 8484, pages 271–284.

Unix Man Pages. Numa Maps - Non Uniform Memory Architecture. `man7.org/linux/man-pages/man7/numa.7.html`.