Provisionamento automatizado de servidores para competições de segurança da informação

Lucas Magalhães[1], Antônio Carlos F. Petri[1], Gabriel de S. Alves[1], Cesar Augusto C. Marcondes[1], Paulo Matias[1]

[1]Departamento de Computação
Universidade Federal de São Carlos (UFSCar)
São Carlos, SP – Brasil

whoisroot@openmailbox.org, {falcaopetri, g4briel.4lves}@gmail.com,
{marcondes, matias}@dc.ufscar.br

Abstract. *Promoting Capture-The-Flag (CTF) competitions requires large operational costs, due to the number of participants scale and problems computing requirements. In problems that involve server exploitation, it is important to provide guarantees that each participant solution do not interfere with others. Thus, to save resources, we propose an automated provisioner that allocates LXD containers to competitors that have achieved a minimum score, in addition we integrate the provisioner to the OpenStack API. Finally, the solution is fully operational at UFSCar Private Cloud and we plan to adopt it during the 2017 Pwn2Win International CTF.*

Resumo. *Promover competições de Capture-The-Flag (CTF) pode envolver um grande custo operacional, pois a complexidade da infraestrutura escala com o número de participantes e de problemas propostos. Em problemas que envolvam explorar servidores, é necessário garantir que as ações de um participante não impactem na resolução de outros. Para economizar recursos, desenvolvemos um provisionador automático que aloca contêineres LXD apenas para competidores que obtenham previamente uma pontuação mínima em problemas que não envolvam explorar servidores, além de integrar-se à API do OpenStack. A solução encontra-se totalmente operacional na nuvem privada da UFSCar, e pretendemos adotá-la na edição de 2017 do CTF internacional Pwn2Win.*

1. Motivação

Um elemento crítico de uma estratégia efetiva de cibersegurança é dispor de pessoas treinadas nas questões mais recentes relacionadas à segurança da informação. Alguns países, como os Estados Unidos, têm se preparado de forma expressiva contra ataques cibernéticos [ABI Research 2015], no entanto, ainda há uma limitação na quantidade e na qualidade dos profissionais, especialmente quando consideramos habilidades mais sofisticadas, tais como segurança por construção, inteligência de ameaças, ou perícia forense após um ataque [Evans and Reeder 2010].

Com o objetivo de reduzir a carência do mercado em profissionais de cibersegurança, empresas, escolas, universidades e instituições militares ao redor do mundo têm promovido competições de "capture a bandeira" (ou CTF, do inglês *Capture-The-Flag*) para incentivar que mais profissionais envolvam-se com tópicos de cibersegurança.

Competições de CTF geralmente são planejadas para proporcionar aos participantes experiência em um amplo espectro de problemas relacionados à segurança de computadores, bem como em técnicas de ataque e defesa que poderiam ser aplicadas em situações reais. Essas competições também podem servir como uma alternativa de baixo custo para recrutar profissionais altamente habilidosos para preencher vagas específicas de emprego. Dentre as habilidades necessárias para participar de CTFs, citamos engenharia reversa, exploração de falhas em binários e em aplicativos *web*, análise forense, criptanálise e programação.

Há dois estilos de competição de CTF: *ataque/defesa* e *jeopardy*. Em competições de *ataque/defesa*, cada equipe recebe uma máquina para defender, conectada a uma rede isolada. A pontuação das equipes é dada com base tanto no sucesso em defender essa máquina como no êxito em atacar as máquinas de outras equipes. Competições de *jeopardy* são mais comuns e geralmente envolvem múltiplas categorias de problemas, cada qual contendo uma variedade de desafios com diferentes pontuações e níveis de dificuldade. Resolver corretamente o problema leva o competidor a obter uma bandeira (*flag*), que deve ser submetida para uma plataforma a fim de computar pontos no placar. As equipes tentam obter a maior pontuação possível ao longo da competição (*e.g.*, num período de 24 horas), mas geralmente não atacam diretamente uma à outra. Ao contrário de uma corrida, esse estilo de jogo estimula as equipes a dedicar tempo para estudar os desafios, e prioriza a quantidade de submissões corretas em vez da velocidade com que são submetidas.

Dentre os desafios propostos em competições de *jeopardy*, há um subconjunto que disponibiliza um arquivo para que os participantes copiem e resolvam o problema localmente em seus computadores. Geralmente enquadram-se nesse grupo os desafios das categorias de engenharia reversa, análise forense e criptanálise. Muitas vezes, há também um subconjunto de problemas que, apesar de requerer ataques a um servidor, não solicitam que o participante tome controle por completo nem altere o estado desse servidor. Por exemplo, para resolver alguns tipos de problema de exploração de falhas em aplicativos *web* vulneráveis a injeção de SQL (*Structured Query Language*), basta realizar consultas somente de leitura (*e.g.,* `SELECT`). Nesse caso, é possível configurar o servidor de forma que a base de dados mantenha-se congelada e não possa ser alterada pelos participantes.

No entanto, a imensa maioria dos problemas de exploração de binários e de aplicativos *web* exige que os participantes executem ações que alteram o estado do servidor ao longo do processo de resolução. Nesses casos, é necessário prover uma infraestrutura separada para cada equipe, a fim de impedir que as demais equipes causem indisponibilidade ou tomem qualquer ação que dificulte a resolução do problema. Por exemplo, em um desafio que envolva tomar controle por completo de um servidor compartilhado entre diversas equipes, a primeira equipe a resolver o problema poderia remover as bandeiras desse servidor, impedindo que outras equipes concluíssem a resolução. A interferência entre diversas equipes também poderia ocorrer de forma acidental, pela simples disputa entre acessos de escrita originados por equipes distintas de forma concorrente.

Para resolver esse problema, diversas competições têm adotado uma infraestrutura baseada em contêineres, na qual cada equipe recebe um contêiner diferente para um determinado desafio. Contêineres são alternativas leves às máquinas virtuais que, no caso do Linux, são implementadas usando os recursos de espaço de nomes e de grupos de controle do sistema operacional. Apesar do núcleo do sistema operacional apresentar uma área de ataque maior que a de um virtualizador, essa tecnologia vem se mostrando

competitiva em diversos cenários de aplicação [Bernstein 2014].

2. Proposta e Principais Funcionalidades

Este trabalho propõe uma ferramenta de software livre capaz de provisionar uma infraestrutura baseada em contêineres para uma competição de CTF em uma nuvem OpenStack pública ou privada disponibilizando, ainda, as seguintes inovações:

- Nossa ferramenta gerencia contêineres sem depender de soluções internamente complexas, como a plataforma Kubernetes. Apesar de estável e de fácil utilização, a plataforma Kubernetes tem demonstrado possuir uma área de ataque demasiadamente ampla para ser adotada em CTFs nos quais os competidores possam vir a tomar controle por completo de alguns dos contêineres. Em competições nas quais foi adotada, ocorreram problemas devido a configurações padrão inseguras, tais como permissão de acesso à API de dentro dos contêineres [Eastes 2017]. Apesar de existirem esforços recentes em prover recursos e orientações de segurança para a plataforma Kubernetes, optamos por uma solução mais simples e menos sujeita a erros. Gerenciamos os servidores por meio de comandos enviados via SSH (*Secure Shell*) e utilizamos somente contêineres providos pelo LXD, projeto cujo foco é justamente ser seguro por padrão, fornecendo uma alternativa mais leve a virtualizadores [Ernst et al. 2016, Ferreira 2017]. Com isso, perdemos alguns recursos da plataforma Kubernetes, como o provimento de infraestruturas com alta disponibilidade (por meio de réplicas do nó mestre). No entanto, argumentamos que esse recurso não é tão importante em CTFs, podendo ser substituído pelo monitoramento da infraestrutura.
- Provisionamos contêineres apenas para as equipes que atingirem uma determinada pontuação mínima nos problemas que não dependam de contêineres, ou seja, aqueles que possam ser resolvidos pelos participantes localmente em seus computadores, ou que tenham sido elaborados de forma a garantir que todas as operações efetuadas em um servidor sejam somente de leitura. Desta forma, economizamos recursos computacionais que seriam alocados para equipes que se inscreveram mas não possuem a intenção de participar seriamente da competição. CTFs divulgados no sítio CTFtime[1], que reúne e avalia por pares diversas competições internacionais, costumam receber centenas de inscrições de equipes. Em edições anteriores da competição Pwn2Win, observamos que apenas pouco mais da metade das equipes inscritas chega a resolver ao menos um dos problemas propostos [Bertochi 2016].
- Nosso provisionador integra-se à *NIZKCTF*, uma plataforma abertamente auditável e resistente a ataques contra seu mecanismo de placar [Matias et al. 2017]. O provisionador monitora continuamente o placar da plataforma *NIZKCTF* para determinar se uma equipe possui pontuação mínima para ter acesso aos contêineres. Além disso, contêineres associados a problemas que já tenham sido resolvidos por uma equipe são automaticamente destruídos, por não serem mais necessários.

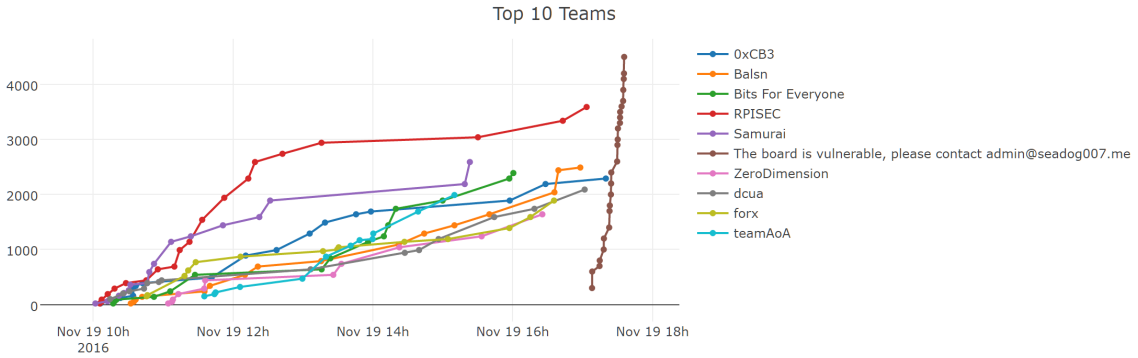
3. Importância da Segurança da Infraestrutura de CTFs

Uma preocupação constante para organizadores de CTFs é proteger de ataques a infraestrutura da competição, que está sujeita às mesmas falhas de segurança que qualquer

[1] `https://ctftime.org`

sistema computacional. Devido à elevada competitividade entre os participantes, é comum encontrar equipes atacando a infraestrutura em vez dos desafios propostos pela competição.

Um exemplo dessa situação ocorreu durante o RC3 CTF de 2016. Por um momento durante essa competição, uma equipe denominada "*The board is vulnerable, please contact admin@seadog007.me*" apareceu no placar com 4500 pontos [Yu 2016]. A Figura 1 mostra o registro desse fato. Felizmente, a intenção do participante que explorou a falha era apenas reportar a vulnerabilidade para os administradores. Entretanto, era perfeitamente possível explorá-la para fins maliciosos.



Place	Team	Score
1	The board is vulnerable, please contact admin@seadog007.me	4500
2	RPISEC	3590
3	Samurai	2590

Figura 1. Durante o RC3 CTF de 2016, um competidor explorou falhas no mecanismo de placar para reportar a vulnerabilidade aos administradores da competição.

Um dos componentes de uma competição de CTF mais visados em ataques é a plataforma central que recebe submissões de respostas (bandeiras) e gerencia o placar. Visando proteger esse mecanismo nas competições em que for adotada, a ferramenta apresentada neste artigo foi desenvolvida para integrar-se à *NIZKCTF*, uma plataforma para CTFs que propomos recentemente [Matias et al. 2017], que baseia sua segurança em provas criptográficas em vez de confiar na segurança do software ou dos servidores que o hospedam. Em lugar de submeter diretamente as respostas (bandeiras) para a plataforma, o competidor envia uma prova de conhecimento zero de que obteve a resposta correta. Essa prova criptográfica não contém nenhuma informação que possa auxiliar outras equipes a resolver o problema, e é tornada pública para que qualquer observador externo possa auditar a competição. A Figura 2 ilustra a arquitetura geral da implementação da plataforma *NIZKCTF*, que utiliza somente serviços disponibilizados gratuitamente (até limites da ordem de 1 milhão de requisições) pela nuvem pública da Amazon.

Além da preocupação em adotar uma plataforma central segura, a ferramenta proposta neste artigo foi projetada para provisionar uma infraestrutura que resista aos modelos de adversário descritos ao final da seção a seguir.

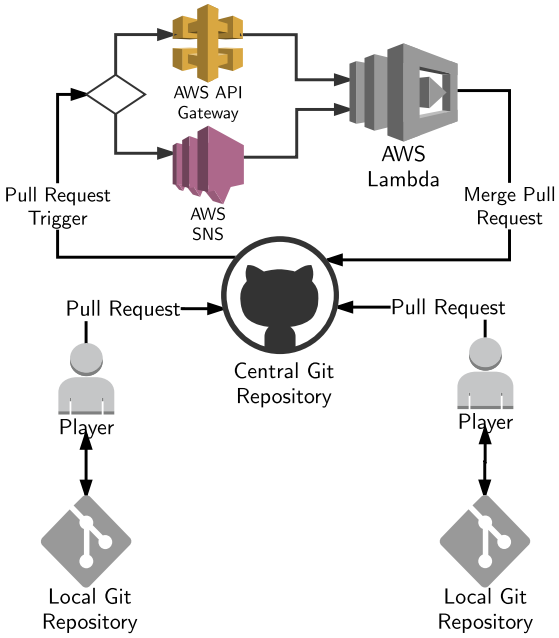


Figura 2. Visão geral da plataforma *NIZKCTF*, à qual o provisionador integra-se. Os competidores modificam repositórios Git locais e criam *pull requests* para o repositório central. Uma função *AWS Lambda* é disparada para integrar as alterações ao repositório central caso as mudanças sejam válidas.

4. Infraestrutura Provisionada e Modelos de Adversário

A Figura 3 apresenta a infraestrutura provisionada pela solução proposta neste artigo. Como forma de provisionamento inicial, utilizamos a ferramenta Ansible para preparar um certo número de máquinas virtuais para receber contêineres de desafios, além de uma máquina virtual específica para abrigar contêineres de VPN (*Virtual Private Network*).

Na rede gerenciada pelo componente Neutron do OpenStack, a ferramenta proposta por esse artigo aloca uma *vlan* para cada equipe que atingir a pontuação mínima necessária para ter acesso aos contêineres. Todas as máquinas virtuais possuem acesso a todas as *vlans*, porém internamente a essas máquinas conectamos cada *vlan* apenas ao contêiner alocado para a respectiva equipe.

Os contêineres de VPN são configurados com certificados TLS (*Transport Layer Security*) emitidos por uma autoridade comum, gerenciada pela própria ferramenta. Os times acessam essas instâncias por meio de um roteamento NAT (*Network Address Translation*) reverso. Cada equipe recebe o número da porta que é internamente roteada para seu contêiner de VPN, juntamente com um usuário e uma senha aleatória para conexão.

Contêineres contendo um mesmo desafio são provisionados sempre na mesma máquina virtual. Como utilizamos ZFS para armazenar os dados dos contêineres, essa organização permite aproveitar ao máximo os recursos de cópia na escrita (*copy-on-write*) desse sistema de arquivos. Ao provisionar um desafio para uma nova equipe, os blocos de armazenamento não são clonados, apenas referenciados. No decorrer da competição, apenas blocos que desviem do conteúdo original causarão um aumento do espaço em disco ocupado na máquina virtual.

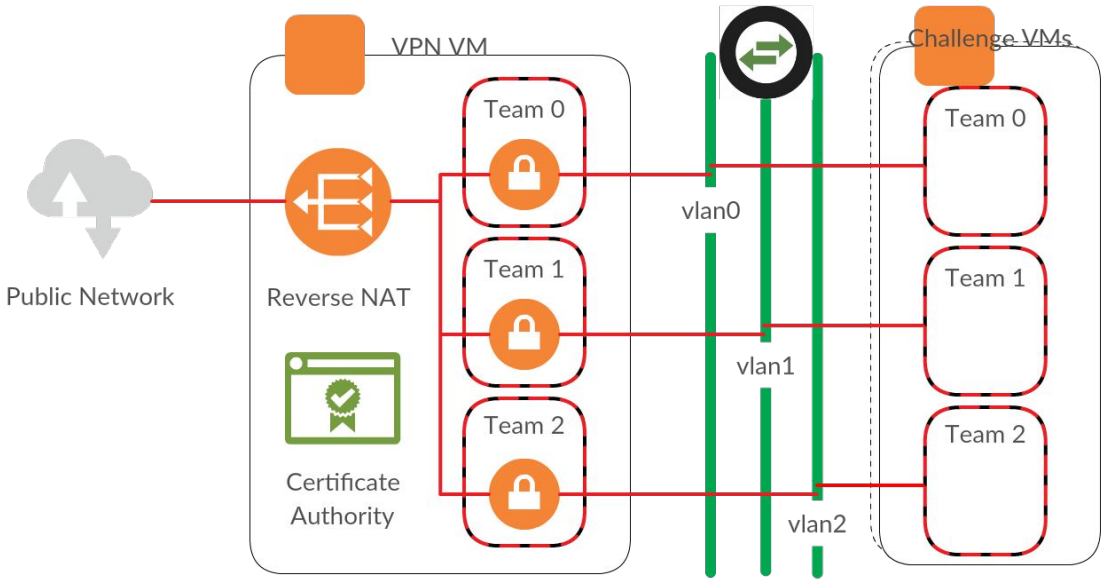


Figura 3. Infraestrutura provisionada pela ferramenta proposta. Cada time é alocado em um contêiner distinto, dentro de máquinas virtuais diferentes para cada problema. As equipes acessam a rede interna da competição via VPN.

Além disso, a organização proposta provê propriedades interessantes de segurança quando adotados os seguintes modelos de adversário:

- Um adversário que possua acesso a uma falha de dia zero (*zero day*) que permita escapar de contêineres, e pretenda obter as respostas (bandeiras) de problemas que não tenha resolvido: Neste caso, o adversário não conseguirá escapar de um problema para outro, assumindo que este não tenha acesso a uma falha de dia zero do virtualizador.
- Um adversário que possua acesso a uma falha de dia zero (*zero day*) que permita escapar de contêineres, e pretenda apagar ou corromper dados dos contêineres de outros times: Neste caso, o adversário precisa ao menos começar a resolver cada um dos problemas antes de perpetuar o ataque. Caso tenha obtido acesso para executar código remoto em apenas uma das máquinas virtuais, somente conseguirá explorar a falha de dia zero nessa máquina.

5. Arquitetura da Ferramenta

O processo principal da ferramenta é implementado por um único *script* escrito em linguagem Python. Esse *script* monitora constantemente o arquivo de submissões aceitas gerado pela plataforma *NIZKCTF* e, por meio da função `compute_target_state`, calcula o estado final desejado, ou seja, quais contêineres devem estar alocados em cada máquina virtual, dada a lista de desafios resolvidos por cada time.

Uma vez calculado o estado final, o *script* executa as seguintes funções, em ordem:

- `start_vms`: assegura que as máquinas virtuais que abriguem contêineres que precisem estar disponíveis estejam ligadas;
- `handle_container_transition`: assegura que os contêineres que precisem estar disponíveis existam dentro das máquinas virtuais, e destrói contêineres que não precisem estar disponíveis;

- `stop_idle_vms`: desliga máquina virtuais que não estejam abrigando nenhum contêiner.

Cada transição (criação ou destruição) de contêiner é efetuada por um *script* em linguagem *shell* previamente instalado pela ferramenta Ansible em cada máquina virtual. A ferramenta chama esses *scripts* por meio de comandos SSH enviados com o auxílio da biblioteca Paramiko.

O *script* que provisiona o contêiner de VPN é especial pois configura as credenciais de acesso da equipe à rede interna da competição e, portanto, precisa comunicá-las de volta à equipe. Para isso, o provisionador adiciona a um repositório Git uma mensagem encriptada com a chave pública da equipe, permitindo que apenas a equipe a que se destina tenha acesso ao conteúdo, sem exigir que o provisionador tenha acesso a contatos diretos ou informações da equipe que não sejam do conhecimento de todos que acompanham a competição.

Todas as operações do provisionador são executadas em *thread pools*, com o objetivo de alcançar melhor desempenho (reduzindo as esperas de entrada/saída) e de permitir impor tempos máximos de execução (*timeouts*).

6. Homologação

A ferramenta proposta neste artigo foi previamente homologada, em pequena escala, em uma competição promovida durante a Semana de Computação (SeComp) do Departamento de Computação da UFSCar.

Durante essa competição, 4 equipes alcançaram pontuação suficiente para receberem acesso aos contêineres. Foram observados apenas problemas pontuais na solução de provisionamento aqui proposta, tais como a inadequação da configuração de *keepalive* do OpenVPN para a rede de alguns dos participantes que estavam atrás de roteadores NAT. Todos os problemas observados puderam ser corrigidos ainda no decorrer da competição.

7. Disponibilidade

O provisionador e projetos correlacionados estão disponíveis como software livre sob a licença MIT. Todos os códigos e documentação estão disponíveis nos repositórios da competição Pwn2Win no GitHub: `https://github.com/pwn2winctf`.

- Documentação de instalação da plataforma *NIZKCTF* (pré-requisito para realizar um CTF com o provisionador): `https://github.com/pwn2winctf/nizkctf-tutorial/blob/master/GitHub.md`
- Documentação de instalação do provisionador: `https://github.com/pwn2winctf/NIZKCTF-provisioning/blob/master/README.md`

8. Requisitos de Hardware

O provisionador foi testado em uma máquina virtual com as seguintes características:

- 1 núcleo de CPU de 2 GHz
- 2 GB de memória RAM
- 5 GB de disco

9. Demonstração Planejada para o SBSEG

Para o Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg), pretendemos instalar o provisionador proposto neste artigo em um computador pessoal (PC) ou *laptop* com acesso à Internet a ser exposto no SBSEG.

Configuraremos o provisionador para monitorar o repositório GitHub controlado por uma instância da plataforma *NIZKCTF* previamente configurada na Amazon. Posteriormente, disponibilizaremos alguns desafios para que os participantes do Simpósio possam simular sua participação em uma competição. Configuraremos o provisionador para instanciar esses desafios em máquinas virtuais na nuvem da UFSCar. Em outras palavras, o computador fisicamente exposto na SBSEG apenas controlará máquinas virtuais hospedadas em uma infraestrutura externa.

Alguns dos desafios terão a resposta (bandeira) incluída em seu próprio enunciado, a fim de permitir que mesmo aqueles que não tenham à disposição tempo para estudar e resolver um problema de CTF possam avaliar o funcionamento do provisionador. Desta forma, ao resolver ao menos um desafio que não tenha um servidor atrelado a si, o expectador da demonstração receberá credenciais para acessar a VPN hospedada na nuvem da UFSCar. Em seguida, ao resolver problemas atrelados a servidores, observará que os contêineres correspondentes são destruídos, e passa a não ser mais possível acessá-los.

Referências

ABI Research (2015). Global cybersecurity index & cyberwellness profiles. Technical report, International Telecommunications Union, Geneva, CH.

Bernstein, D. (2014). Containers and cloud: From LXC to Docker to Kubernetes. *IEEE Cloud Computing*, 1(3):81–84.

Bertochi, A. (2016). Pwn2Win CTF 2016 – Bastidores. https://ctf-br.org/2016/04/pwn2win-ctf-2016-bastidores.

Eastes, B. (2017). Capturing all the flags in BSidesSF CTF by pwning our infrastructure. https://hackernoon.com/capturing-all-the-flags-in-bsidessf-ctf-by-pwning-our-infrastructure-3570b99b4dd0.

Ernst, D., Bermbach, D., and Tai, S. (2016). Understanding the container ecosystem: A taxonomy of building blocks for container lifecycle and cluster management. In *IEEE Second International Workshop on Container Technologies and Container Clouds*, Berlin, Germany. IEEE.

Evans, K. and Reeder, F. (2010). A human capital crisis in cybersecurity: Technical proficiency matters. Technical report, Center for Strategic and International Studies, Washington, DC, USA.

Ferreira, U. J. S. (2017). Análise de tecnologias de virtualização e hardware de baixo custo para infraestutura de nuvem de pequeno porte. https://memoria.ifrn.edu.br/handle/1044/940.

Matias, P., Barbosa, P., Cardoso, T., Mariano, D., and Aranha, D. (2017). NIZKCTF: A non-interactive zero-knowledge capture the flag platform. https://arxiv.org/abs/1708.05844.

Yu, J. (2016). Seadog GitHub repository: RC3-CTF-2016-scoreboard. https://github.com/seadog007/RC3-CTF-2016-scoreboard.