Economic Denial of Sustainability Attacks on Kubernetes

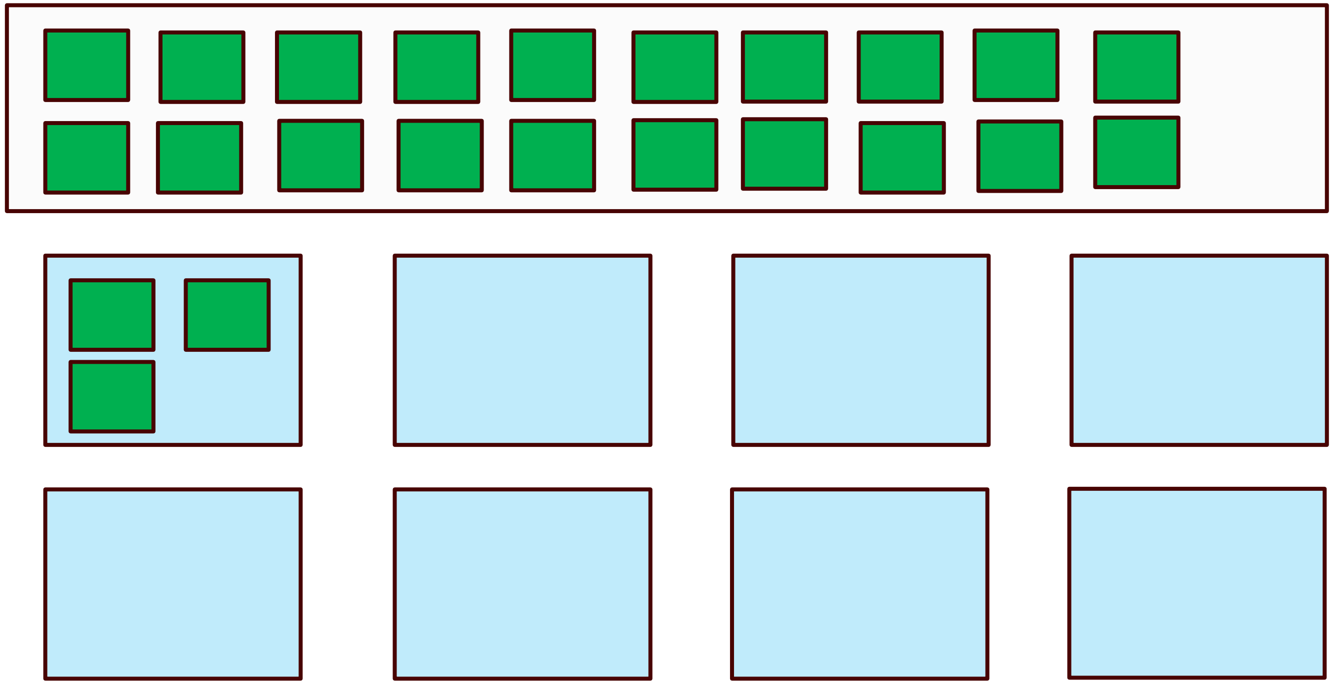
JONATHAN CHAMBERLAIN

PRESENTATION FOR SEMINAR ON PRACTICAL SECURITY 2023-08-10

Overview

- The cloud really is “someone else’s computer”
  - Business model: server providers concern themselves with maintaining hardware, application owners need only focus on the software.
  - Allows for rapid capacity expansion: lease additional compute nodes as needed.
  - How to determine when to scale?

Overview

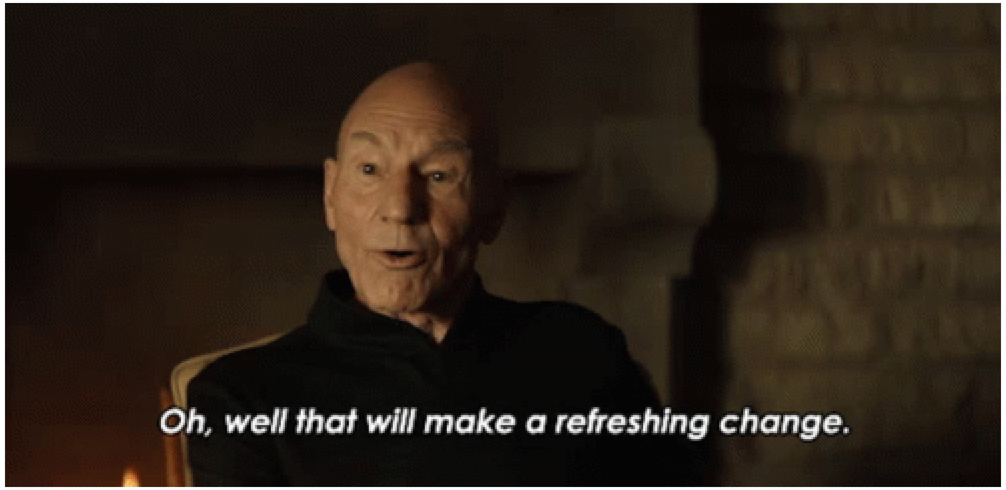


Overview

- Autoscaling is useful for DDoS defense.
  - Having autoscaling enabled is considered a security best practice by Amazon and others for this very reason [1].
  - Compute nodes are not free.
  - Thus, the defense is itself open to exploitation.
- This can result in an Economic Denial of Sustainability (EDoS) scenario.

Overview

- EDoS in this context is accomplished by flooding the system with excess traffic to forcibly trigger the autoscaling mechanisms.

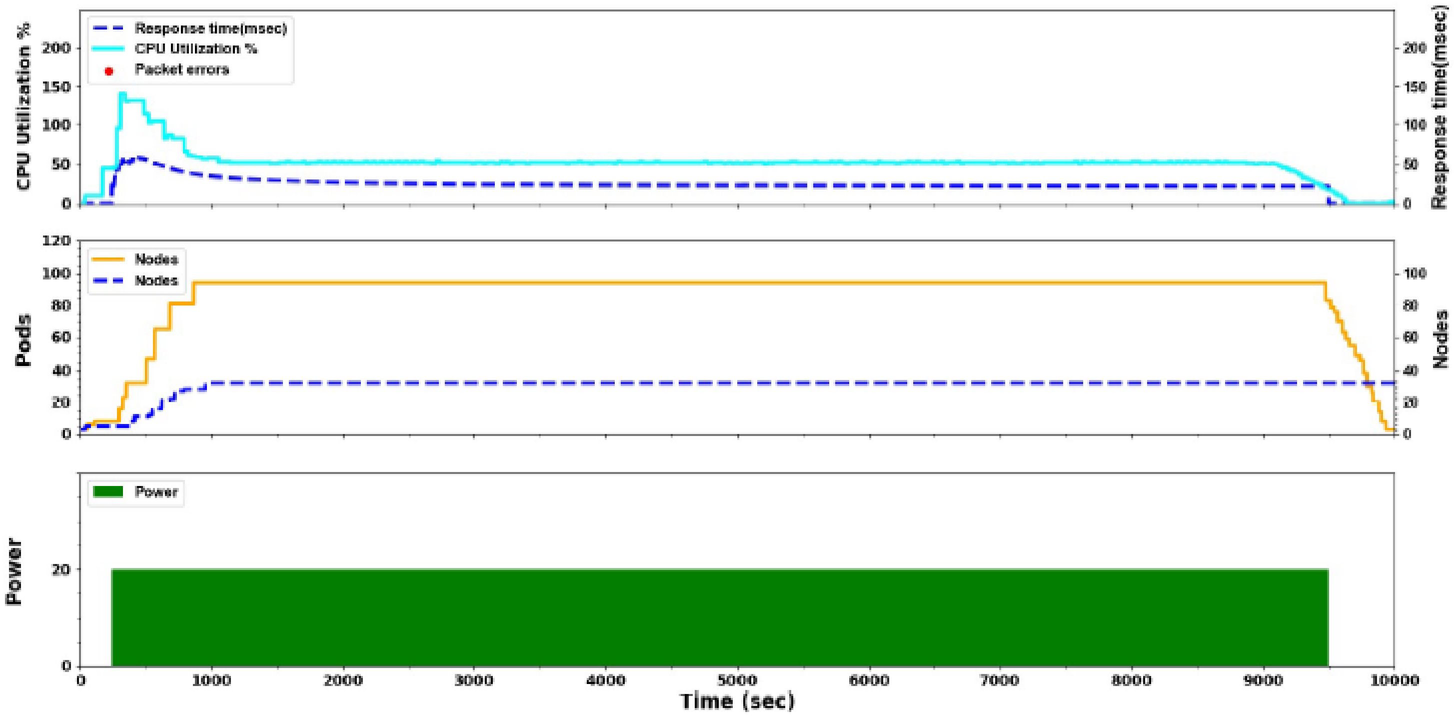


Yo-Yo Attacks in Kubernetes

- A particular class of EDoS attack, the Yo-Yo, was demonstrated as a proof of concept by Ben David and Bremler-Barr. [2]
- The key finding is that with Yo-Yo, nearly the same economic cost is imposed on the target without the need to send continuous traffic.

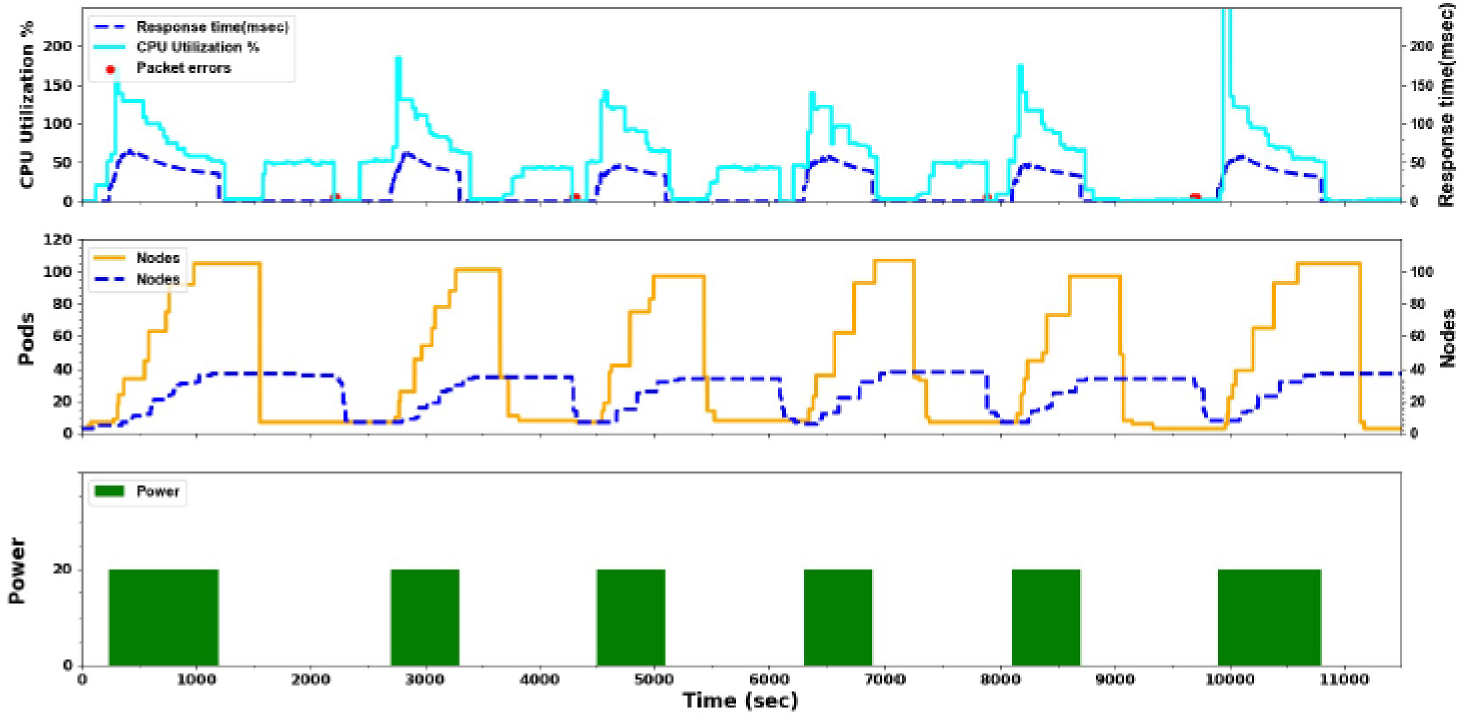
Yo-Yo Attacks in Kubernetes

- Consider an example DDoS attack on Kubernetes infrastructure [2]



Yo-Yo Attacks in Kubernetes

- Contrast with the comparable Yo-Yo attack [2]



Yo-Yo Attacks in Kubernetes

- While these experiments were done in a controlled environment, the underlying attack structure is increasingly common.
  - In 2021, wave attacks accounted for approximately half of all DoS style attacks [3]
  - However, a sample of corporate attendees at a Red Hat Research Days talk indicated that they had no plan for dealing with such attacks [4].

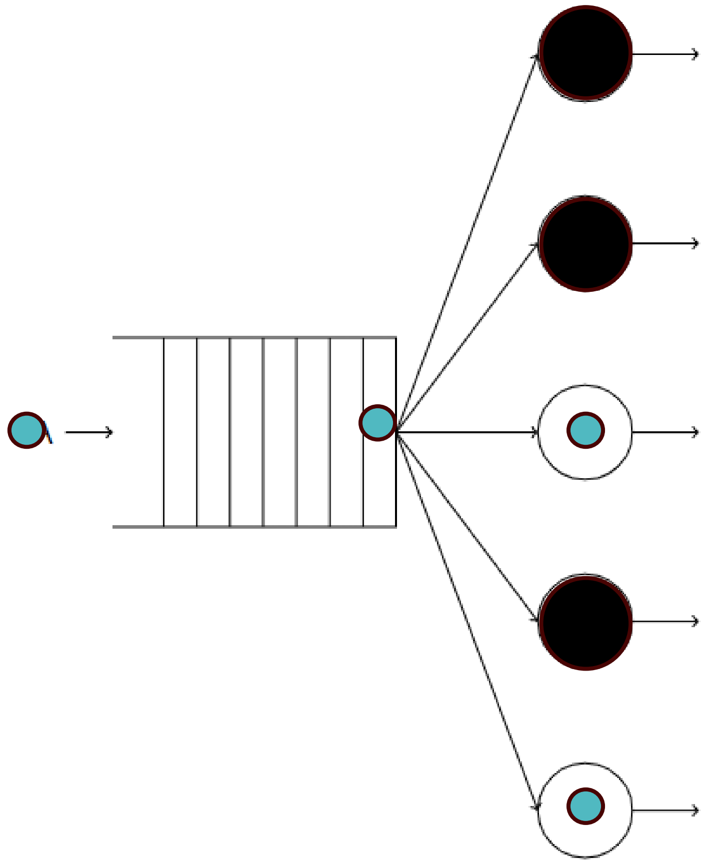
Yo-Yo Attacks in Kubernetes

- Further, there is a question as to whether the Yo-Yo is really the “best” that can be done:
  - The attack as structured in [2] is not randomized in any way
  - In [2], scaling only occurs as a result of adversarial traffic
  - Attack/rest periods determined solely by scaling time assuming traffic goes from some constant level of normal traffic, to maximum adversarial traffic, back to normal traffic levels again

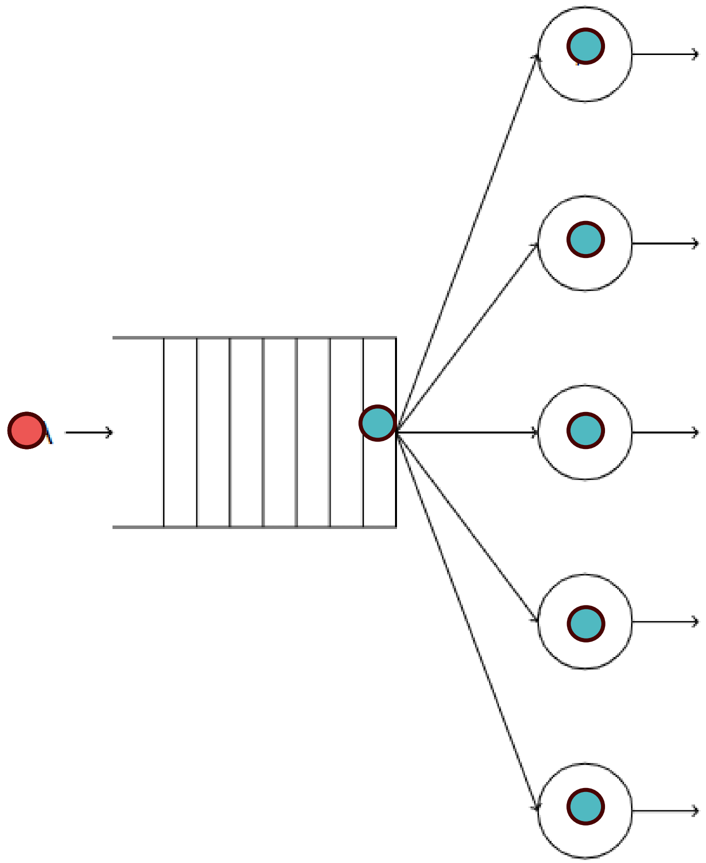
Improving on the Yo-Yo Attack

- By default, the Cluster Autoscaler software checks every 10 seconds to determine whether the number of nodes need to be reallocated [5].
- Our hypothesis: an attacker can exploit timings/system states to inject just enough traffic to keep the system at a higher number of nodes for cheaper than the wave attack.

Improving on the Yo-Yo Attack



Improving on the Yo-Yo Attack



Improving on the Yo-Yo Attack

- In addition, while the Cluster Autoscaler is itself open sourced on GitHub, the current deployment is through managed cloud providers [5].
  - Thus, certain settings become a “black box” which cannot be updated by the application owner.
- Our current efforts are focused on two areas:
  - Developing a Game Theoretic model
  - Running experiments on OpenShift utilizing the NERC

Game Theoretic Approach

- Essentially, want a quantifiable and provably optimal policy which can be taken by the attacker, given assumptions on the system settings.
  - Allows for evaluation of how system changes impact attacker strategy
  - Can design defense around expected attacker behavior.
- Accomplish this through Markov Decision Processes (MDPs)

Game Theoretic Approach

- MDPs are described by:
  - States, in this case (m,n) where m is the autoscaling metric of interest, and n is the number of active nodes
  - Actions, in this case to add a node, remove a node, or keep the number of nodes
  - Costs/rewards, e.g. cost of running extra nodes, cost of activating/deactivating nodes, refunds to customers for SLA violations, perceived costs of over/under provisioning

Game Theoretic Approach

- At each step, the action which maximizes reward/minimizes cost is taken.
- Over time, the actions taken define a policy, which can be expressed as a Markov chain.
- In our case, this will yield the thresholds for activating/deactivating the next node.
  - If the attacker can gleam the state of the system at any given time, this information allows for better attack planning.
- Solving MDPs is complex – currently adapting python code to accomplish this for our case(s).

Experiments on OpenShift

- In addition and in parallel to the theoretic approach, we intend to run experiments on a NERC OpenShift instance through creating a cluster of our own.
- Ideally, this enables full control over the relevant settings, rather than some settings be confined to provider fiat, as in [2]

Experiments on OpenShift

- In particular, the Node scale-up/scale-down can be controlled through Cluster Autoscaler [5]

Parameter	Definition	Configuration given by	Value
r	Average requests rate per second of legitimate clients	System usage	
N_p	Initial number of Pods	System administrator	4
N_n	Initial number of Nodes		4
R	Number of Pods per Node		3
$I^p_{up} \backslash I^p_{down}$	Threshold interval for scale-up and scale-down for a Pod		1min\5min
$I^n_{up} \backslash I^n_{down}$	Threshold interval for scale-up and scale-down for a Node		10sec\10min
$W^p_{up} \backslash W^p_{down}$	Warming time of scale-up and scale-down for a Pod	Kubernetes infrastructure	30sec\5sec
$W^n_{up} \backslash W^n_{down}$	Warming time of scale-up and scale-down for a Node		2min\2min
k	The power of the attack	Attacker	
n	Number of attack cycles		
T	Cycle duration		10\20
$t_{on} \backslash t_{off}$	Time of *on-attack* phase and *off-attack* phase. $T = t_{on} + t_{off}$		

Experiments on OpenShift

- The main roadblock at present is that OpenShift magnum, which will allow for integration with Cluster Autoscaler, is not currently production code
  - Scheduled for October release as part of OpenShift 2023.2 update
- As a result, we are currently working to try and simulate cluster behavior via other means, as well as running experiments with minikube to determine behavior when the number of nodes is fixed.

Conclusions

- Wave (a/k/a burst) attacks are an emerging threat to cloud infrastructure.
- Tools utilized to prevent such attacks from creating a DDoS scenario can be weaponized to create an EDoS scenario instead.
- Proof of concept in the form of periodic Yo-Yo attacks exists.
- Arguably not the most sophisticated attack possible, current research efforts are working towards proving this.
  - Goal is ultimately to develop better detection and defense capabilities against such attacks.

References

- [1] Amazon Web Services, “AWS Best Practices for DDoS Resiliency,” 2021. https://d0.awsstatic.com/whitepapers/Security/DDoS White Paper.pdf
- [2] R. Ben David, and A. Bremler Barr. "Kubernetes autoscaling: Yoyo attack vulnerability and mitigation." *arXiv preprint arXiv:2105.00542* (2021).
- [3] Imperva. *2021 Cyberdefense Threat Report*. 2021. https://www.imperva.com/resources/resource-library/reports/2021-cyberthreat-defense-report/
- [4] A. Bremler-Barr and M. Czeizler. *Research Days 2023: Cloud Auto-scaling Mechanism Under DDoS Attacks: Yo-Yo Attack and Tandem Attack*. Red Hat. 17 March 2023. https://www.youtube.com/watch?v=_KTfY5NBIB8
- [5] Kubernetes. Cluster Autoscaler. July 2023. https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler