Chaos Engineering and the Disruption Platform

ARMS
Applied Resilience for Mission Systems

Applied Resilience for Mission Systems

Bich Vu and Orton Huang

Overview

The DoD defines resilience as "the ability of the collection of systems to support the function necessary for mission success in spite of hostile action or under adverse conditions"[1]. With this in mind, do you know how your system performs under adverse conditions or when disruptions occur? There are numerous tests to make sure systems work properly, but are there tests to ensure that an operational system responds or recovers as expected when things go wrong?

By adopting practices like Chaos Engineering, we can make our systems more resilient by breaking things in operations, and learning to engineer and plan for disruptions. Chaos Engineering is "the discipline of experimenting on a system in order to build confidence in the system's capability to withstand turbulent conditions in production"[2].

Our Disruption Platform and process are based on the Principles of Chaos Engineering[2] and aim to systematically apply these principles to DoD mission systems. With this, we can improve our understand of how the system performs under different disruptions and learn how to make the mission system more resilient, both in terms of its capabilities and by practicing organizational processes.

Need

Disruptions due to internal or external causes are inevitable and can occur when we least expect it. We need to improve our systems' ability to withstand and respond to these disruptions. On the morning of 28 February 2017, AWS S3 servers were accidentally brought down due to a human error[3]. This caused an outage for more than four hours and interrupted the services of many businesses, such as Instagram and American Airlines, but not Netflix[4]. Netflix was able to overcome this disruption due their Chaos Engineering mindset and learning from their previous failures[3].

To provide increasingly advanced functionality, the complexity of our mission systems has grown significantly. It has also become increasingly difficult to understand our software components and how they interact with each other. Along these lines, it is difficult to correctly design and architect resiliency up front due to these complexities and the potential for undesired and unexpected emergent behaviors, which may not show up until the system is in operations.

Additionally, a process to understand how the system performs under disruptions is not common for DoD systems, and there is no systematic way to learn about issues during development. With the fear of things breaking and no quick way to recover, it is even more difficult to test in production. Also, there is a need to practice a mission system's response when something does go wrong, both from technical and organizational perspectives.

Approach

Our approach focuses on learning how to apply Chaos Engineering in the DoD to improve the resiliency of its mission systems. We practice disrupting the system to understand its resilient properties and to learn how to improve the system's technical capabilities and the organizational processes. To this end, we have developed the Disruption Platform, which provides capability to:

- Practice Chaos Engineering in production/operation
- Practice disruptions as far left of production (during development and integration) as possible
- Practice an organization's response to disruptions

To gain utility from a platform that generates disruptions, we must be deliberate in what we are trying to learn from the system and follow a systematic process. For our case, we utilize the following process shown in Figure 1.

[1] DoD. Dec 2015. JCIDS Manual.

[2] Chaos Community, May 2018. "Principles of Chaos Engineering." https://principlesofchaos.org

[3] C. Churchey. March 2017. "Why Netflix didn't sink when Amazon S3 went down." https://www.networkworld.com/article/3178076/why-netflix-didnt-sink-when-amazon-s3-went-down.html

[4] R. Whitwam. February 2017, "Amazon S3 Outage Has Broken A Large Chunk Of The Internet." https://www.forbes.com/sites/ryanwhitwam/2017/02/28/amazon-s3-outage-has-broken-a-large-chunk-of-the-internet

For more information contact:
ARMS@LL.MIT.EDU

LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Chaos Engineering and the Disruption Platform

ARMS
Applied Resilience for Mission Systems

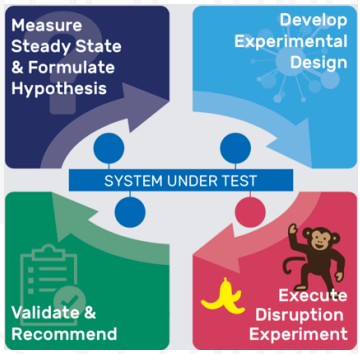

Figure 1. The Disruption Platform Process.

Measure Steady State & Formulate Hypothesis. We need to understand how the system performs normally. This requires having an appropriate level of observability. It is also helpful to have a simple metric to capture the "pulse" of your system[5]; this could be different depending on the focus of the experiment. From there, we have a reference baseline and a basis to formulate hypotheses we want to test. A hypothesis should be well defined and can follow the following format:

"The <mission performance> will <persist/degrade/ fail> when <disruption condition> occurs."

Develop Experimental Design. Next, we develop our experiment with the following considerations:

- What is the targeted node(s) and/or service(s)?
- What is the blast radius of the disruption?
- What is the disruption type and severity to perform on the target?
- What is the duration and cadence of the disruption?

If there are new disruption types to consider to properly test the hypothesis, they should be implemented and tested at this point.

Execute Disruption Experiment. Next, we execute the disruptions. It is essential to monitor how the disruptions impact the system, especially when it is a new disruption, and experiments can be stopped early if findings are observed or if the system is impacted more than expected.

Validate & Recommend. Last, we can use our observation capability to verify the accuracy of the hypothesis. We can analyze the outputs to document whether there are any areas of improvements, make recommendations from those observations where appropriate, or design new experiments. This is an iterative process to learn more about the system and its weaknesses while making improvements and/or adjustments to mitigate issues in the process.

At this point, it is good to evaluate whether we need more iterations on the process to get actionable findings.

Our Disruption Platform is designed to run both in the cloud and on-premises, and is enabled through the following tools: Ansible, Docker, Python, AWS CLI, command line tools, and the Chaos Toolkit. Based on key functionality in our surrogate system, as of this writing, we implemented disruptions of the services (start/stop services/containers/ hosts), the network (delay, loss, corruption, duplication), and the ability to spin up new disruptive services to stress the system.

In the DoD, applying this process can be daunting. We recommend learning to apply this process first during development and integration testing. In operations, where appropriate, spend time coordinating with all stakeholders and start small with a contained blast radius to build confidence and support.

Impact

In applying Chaos Engineering, we are better able to understand how the system performs under stress. This is useful to apply throughout the lifecycle of a mission system (from development to operations). Building a culture of learning from failure provides a forcing function towards developing more resilient software. This practice also provides insights into how the system performs when integrated together and can help investigate undesired emergent behaviors at a time when it is less impactful to the mission. Additionally, it provides an opportunity and mechanism to practice the system and organizational response when failures occur at a time that is not detrimental to the mission.

What Should Builders Do?

- Develop software with the mindset that it will be tested with disruptions both in development and in production
- Adopt a Chaos Engineering mindset throughout the lifecycle of the mission system
- Implement the disruption process as early in the development cycle as possible to learn how to apply it in operations
- Include Chaos Engineering approach in Testing and V&V steps
- Develop and leverage capabilities like the Chaos Monkey, Simian Army, and Disruption Platform to enable this approach
- Leverage tools like the Disruption Platform to exercise your organization's response

[5] Netflix Technology Blog. February 2015. "SPS: the Pulse of Netflix Streaming." http://techblog.netflix.com/2015/02/sps-pulse-of-netflix-streaming.html

This material is based upon work supported by the United States Air Force under Air Force Contract No. FA8702- 15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY