MITRE | Center for Data-Driven Policy

SERIES Number 9

INTELLIGENCE AFTER NEXT

DESIGNING FOR CHAOS LEADS TO RESILIENCY

by Jonathan Rotner and Matt Yetto

Wait, you want me to break my system…on *purpose?*

"Imagine getting a flat tire. Even if you have a spare tire in your trunk, do you know if it is inflated? Do you have the tools to change it? And, most importantly, do you remember how to do it right? One way to make sure you can deal with a flat tire on the freeway, in the rain, in the middle of the night is to poke a hole in your tire once a week in your driveway on a Sunday afternoon and go through the drill of replacing it. This is expensive and time-consuming in the real world, but can be (almost) free and automated in the cloud."[1]

This story comes from Netflix, whose services run on a large, distributed system, and serve customers across the globe at all hours of the day. Such a complex, interconnected and interdependent structure means something can always go wrong–hardware will fail, software will be imperfect, the electrical grid will act unexpectedly, users will surge without warning. To get ahead of the unpredicted and the undesired, Netflix introduced failures, over and over again, through system development and into operations. This practice is called chaos engineering.[2]

The government runs similar services keeping the citizens of the United States safe across many large, distributed systems serving national security professionals across the globe at all hours of the day. In addition to the routine hiccups[3] from complex and interconnected infrastructure, our national security assets are routinely targeted for disruption by others. Introducing chaos engineering tools and practices to secure, on-demand communications networks can ensure that the missions relying on those networks remain resilient even under attack.

Chaos engineering doesn't *create* chaos, but anticipates and reduces it

The former Netflix cloud architect, Adrian Cockcroft, described chaos engineering as "an experiment to ensure that the impact of failures is mitigated."[4] In effect, chaos engineering is a well-defined set of practices, run by experts who design targeted and controlled automated experiments.

Targeted means that each tool is developed to induce a single type of problem. For example, Netflix's Chaos Monkey randomly shuts down server instances, imitating server failures.[5] In fact Netflix had a whole "Simian Army" of tools, each designed to create a single type of problem.[6] Here, chaos engineering tools are targeted to test situations where the system is not expected to fail.

Controlled means that tests occur on live networks, under supervised conditions. Planned disruptions allow specific engineering teams to be notified when they will be on call, which allows them to fix problems during business hours. Intentionally inducing failures motivates developers to write code that is more forgiving to imperfection and flexible to outages, which in turn makes their jobs easier down the line.[7] In addition, creating problems while the system is operational leads to a higher level of confidence that fixes will work when those systems need to operate.[8] Probing for outages in controlled circumstances allows teams to find and strengthen the weakest links in the system.

CHAOS ENGINEERING EMPLOYS SPECIFIC PRACTICES AND TOOLS TO CREATE A SHIFT IN MINDSET.

Chaos engineering can be compared to a defensive driving class conducted on a closed course; drivers are exposed to uncomfortable and scary situations preparing them to control a vehicle in unexpected situations. Likewise, chaos engineering accelerates the exposure of system failures that occur through normal and real-world events allowing engineers and developers to address them proactively.

Chaos engineering makes the system more resilient by proactively finding weaknesses

One tool might look for security weaknesses. According to the National Security Agency *Mitigating Cloud Vulnerabilities* report,[9] the two most common cloud vulnerabilities are misconfiguration[10] and poor access control.[11] For many commercial cloud services, chaos engineering could demonstrate whether systems and resources are configured in a way that checks for these vulnerabilities and therefore increases cyber resiliency for the environment and application.

Another chaos engineering tool might promote more effective code. Returning to the Netflix experience, transitioning to the cloud uncovered a failover problem:[12] when server instances unexpectedly stopped and were replaced, the new server's settings were not configured correctly.[13] In response, Netflix designed Chaos Monkey, which shut down server instances randomly, effectively forcing engineers to design a more resilient and relevant solution in a real-world situation.

Unlike Netflix and other commercial platforms, U.S. government agencies—including those in the intelligence community—are not using chaos testing. Yet, if used well, these kinds of tests can add to the value from existing government test and evaluation (T&E) activities,[14] which are aimed at documenting the conditions under which a system works. The increase in ransomware attacks[15] has revealed that organizations do not adequately test their disaster recovery plans. Chaos tests could be highly effective for exercising continuity of operation plans. The live nature of chaos tests provides further confidence in the application of maturing technologies like the cloud.

Choosing chaos engineering—what to know

Chaos engineering employs specific practices and tools to create a shift in mindset. Chaos engineering assumes and introduces failure, rather than minimalizing and reacting to it. This shift in philosophy can create two important effects. First, code developers and their leadership become more comfortable responding to unanticipated system problems, since chaos engineering practices force them to deal with the unpredictable and undesired. Second, the infrastructure becomes more resilient and fault tolerant by becoming more modular and debuggable.[16] Indeed, as a result of adopting chaos engineering, Netflix saw that "individual components can fail without affecting the availability of the entire system. In effect, [they became] stronger than [their] weakest link."[17]

Before adopting chaos engineering tools and practices, organizations should answer the following five questions:

1. **What is the organization's risk tolerance when it comes to system downtime?**

   Chaos engineering tools yield the best results when running on live systems. Netflix is motivated by profits, and network downtime is a big risk to its bottom line. Chaos engineering fits its need to rely on a network that ensures constant service to all customers without any downtime—in effect, without reduced profits.[18]

   Where commercial industry is motivated to avoid lost revenue and reduced profits, the government is motivated to avoid mission failure.

The consequences for a government system being offline could range from the workforce not being able to do their work, to lost data, to defense systems that temporarily cannot protect the nation.

Of course, long-term system resiliency may offset short-term system downtime. But that tradeoff, as well as options and mechanisms to maintain operations, must be evaluated by each government organization in the context of its own missions.

2. Do long-term cost savings or performance improvements warrant near-term costs?

While chaos testing demonstrates long-term savings, many resiliency techniques are cost multipliers in the short term. Sometimes, the only way to prevent failure is to purchase redundant systems, potentially in different locations. Taking such steps means costs could double, triple, or more. Introducing chaos engineering to development and management teams will also require introductory training to cover its principles and approaches. Finally, chaos engineering tools will cost money to develop or acquire.[19]

Existing chaos engineering tools are not plug and play. Even though companies, resources, and open-source code exist, the software will need to be tailored to the needs and the environment of each government network. Implementing chaos engineering will require hiring and training of chaos engineering experts familiar with both chaos engineering and the operation domain in order to target and tailor tools to the given environment. For example, poorly designed chaos testing may provide intelligence about the network to an adversary, or create or exacerbate a problem where there was none before. Experience—whether previously acquired or learned through mistakes—comes at a cost. However, initial investments could overcome the long-term costs of downtime. In 2014, Gartner estimated network downtime costs companies $300,000 per hour.[20]

Additionally, downtime requires engineers to spend time identifying the cause of, and repairing outages to, the network. There are opportunity costs for developers too—the time it takes to find and fix problems could be spent on new features or higher priority items.[21] Despite these concerns, ultimately, problems—and solutions—could be identified more quickly and at lower cost through chaos engineering approaches.

3. Do other forms of resiliency engineering meet the mission need?

Government organizations already employ approaches that harden the system. They take the form of:

- DevSecOps, which is "an organizational software engineering culture and practice that aims at unifying software development (Dev), security (Sec) and operations (Ops). The main characteristic of DevSecOps is to automate, monitor, and apply security at all phases of the software lifecycle: plan, develop, build, test, release, deliver, deploy, operate, and monitor."[22] Government organizations are beginning to codify these practices; in fact, that definition comes from a document that describes implementation and operational guidance of DevSecOps for the Department of Defense Information Technology (IT) Enterprise.
- Red teaming, which is where the government brings in a team to find and expose vulnerabilities. These can be white-hat hackers, who are experts, often formally certified, who hack to aid an organization without causing harm.[23] [24] Organizations such as Apple[25] and the Department of Defense[26] have hired white-hats or posted rewards for identifying and sharing vulnerabilities.
- Components of chaos engineering are also already part of software development best practices. Best practices usually involve stress testing (driving a service to its breaking point), fuzz testing (use of random or invalid inputs), corner cases (extreme cases within valid ranges), and failure

injection (causing a failure at the application, network, or infrastructure level), which are all components of chaos engineering.

- Cyber expertise and investment wherein the outcomes of certain cyber-attacks can resemble chaos engineering events—the government is already budgeting for $18.8 billion on cybersecurity spending.[27]

Industry best practices include the activities above, and Gartner's 2020 Hype Cycle for Cloud Security advises that chaos engineering is becoming more popular: "The practice is moving beyond innovative early adopters and being utilized in leading enterprises in the financial services and online retail industry. While there continues to be substantial interest in the wider IT community, chaos engineering will eventually find its way to more enterprises over the next few years as many mature their digital initiatives…as companies attempt to build scalable, highly available systems."[28] Chaos engineering makes the system more debuggable and modular, or said another way, problems can be found more quickly and then are easier to isolate. Chaos engineering pushes everyone to design the system with failures in mind. While not mainstream yet, it may be more standard practice soon, and government systems should want to keep pace.

Organizations that choose to add chaos engineering to the "tools in the toolkit" can leverage the enthusiasm and procedures surrounding activities already in place. In fact, chaos engineering tools should be designed to complement and integrate with DevSecOps and other cyber defense efforts.

4. How will vendor contracts require and measure resiliency testing on cloud systems?

The very characteristics that allow for comprehensive automation and DevSecOps implementation create complexity for system monitoring and response. Cloud technology provides the advantages of abstraction—creating a more complex and powerful baseline of code that is hidden from the user, allowing the user to start with more functionality and capability.[29] In the cloud, boundaries between the application and the hidden support are blurred, and this feature can complicate redundancy and recovery.

As a result of abstraction, information technology specialists may not actually know where processes are physically being executed, know what the network topology is, or have insight into the cloud infrastructure software. As an example of the latter case, a cloud service provider may develop and provide a firewall capability.[30] But those that use the firewall have to accept it as it is—it may be impossible to test or affect internal failover[31] or other measures of resiliency. The ability to test resiliency is further limited as more offerings move from Infrastructure-as-a-Service (IaaS) to Software-as-a-Service (SaaS).[32]

The monitoring and response complexities create additional challenges when establishing vendor contracts. Defining resiliency when setting vendor requirements can be difficult; for example, how do acquisition personnel set threshold resiliency measurements, or comprehensively list the types of failures that must be tested in a contract? In addition, chaos engineering works on live systems, so discovery of problems will occur late in the contract or after the government has accepted a delivered product. As a result, the government may be too late to ask for changes to the product or the government may be accepting products that do not provide the resiliency or survivability it requires. Therefore, the government will need to determine methods for including chaos engineering metrics into system requirements while also allowing for agility in response to failure discovery.

5. Who will have ownership over, and set approval for, chaos engineering tools?

Chaos engineering tools could probe the entire system architecture, work across local infrastructures, or even dive into how a specific application interacts with the network. Wherever chaos engineering tools are to be used, network owners will require proper approval before allowing those tools onto the network.

If an architecture is limited to a single government organization, approval to operate should come from that organization. But if the network connects different organizations or is maintained by more than one set of owners, then vetting, approval, and maintenance of the chaos engineering tool will need to be coordinated. Therefore the government will have to choose either centralized or decentralized ownership, responsibility, and accountability for chaos engineering practices and responses.

Some government organizations have already approved use of chaos engineering tools. Chaos Monkey has been approved for use by the Department of Veterans Affairs' Enterprise Architecture team.[33] Chaos engineering software approaches are gaining traction within specific groups in the DoD, and the United States Air Force has had a chaos engineering mentality for years, for example "inject[ing] seven different failure configurations, including shifts in the center of gravity and changes in aerodynamic parameters" into a jet during real test flights, in order to test the aircraft's automated responses to highly consequential conditions.[34] Capturing lessons learned and best practices from these applications can help identify the right approach for your organization's chaos engineering practices and responses.

Embracing chaos

After weighing the benefits and risks to chaos engineering, your organization may want to take the next step. A best practice is to start with a pilot program on a single component of the architecture—for example a virtual machine, server, load balancer, or database—outside of a live environment. It is crucial to anticipate and minimize the potential impact an injected failure may have. Asking questions like who might be affected and what functionality is lost will help. Earn trust by communicating what is happening, giving ample time for others' input, and sharing results or inviting others to participate in exercises, for example, cybersecurity teams. Set up metrics for success before trying out any given tool.

CHAOS ENGINEERING PUSHES EVERYONE TO DESIGN THE SYSTEM WITH FAILURES IN MIND.

Some metrics to consider are: latency,[35] service error rate,[36] failover time,[37] auto scaling duration,[38] and resource saturation of the central processing unit and memory.[39] Setting clear expectations for the team based on mission requirements and emphasizing that chaos engineering is designed to uncover the operational impact of a failure event and to give teams the tools and confidence to respond will help to build trust in the process.

Finally, having some way to pause the experiment and quickly return to normal operation is prudent, particularly as an organization is testing chaos engineering practices.[40]

Once successful, the team can, and will, grow bolder. Experiments can then be conducted on multiple components of the architecture, or across different locations. Eventually, chaos engineering tools can move beyond testing on system architecture, and be implemented across networks, or at the application level.

There are many diverse chaos engineering tools, with different strategies for causing chaos, available through several open-source locations. Some examples with links to further information are compiled in the Chaos Engineering Resources below. Choosing the right tools and practices can improve a system's resiliency posture, but chaos engineering is one part of a failure recovery plan.

Chaos engineering resources

Netflix open-source code

- **Netflix's Spinnaker.** This is an open source, multi-cloud continuous delivery platform for releasing software changes with high velocity and confidence. It includes different elements of the original Simian Army, including Chaos Monkey and Conformity Monkey. https://spinnaker.io/
- **Netflix's Chaos Monkey** randomly terminates virtual machine instances and containers that run inside of your production environment. https://netflix.github.io/chaosmonkey/
- **Netflix's Swabbie** (this project is under development). Swabbie automates the cleanup of unused resources such as EBS Volumes and Images. https://github.com/spinnaker/swabbie
- Retired version of Netflix's **Simian Army.** https://github.com/Netflix/SimianArmy

Open-source code for infrastructure, network, and applications

- **Chaos Toolkit.** Chaos Toolkit is a project whose mission is to provide a free, open, and community-driven toolkit and application programming interface (API) to all the various forms of chaos engineering tools that the community needs. https://chaostoolkit.org
- **Litmus** is a toolset to do cloud-native chaos engineering. Litmus provides tools to orchestrate chaos on Kubernetes to help Software Reliability Engineers find weaknesses in their deployments. https://github.com/limuschaos/litmus
- **Shopify's Toxiproxy.** This is a framework for simulating network conditions and proving that your application doesn't have single points of failure. https://github.com/shopify/toxiproxy

Sample commercial vendors[41 42]

- Alibaba Cloud
- Amazon AWS
- Bloomberg
- ChaosIQ
- Gremlin
- Microsoft
- Netflix
- OpenESB
- Verica
- VMware

Sample reading material

- **Chaos Engineering: the art of breaking things purposefully** (a multipart series) https://medium.com/the-cloud-architect/chaos-engineering-ab0cc9fbd12a
- **Tools for Creating Chaos Outside of AWS** https://www.gremlin.com/chaos-monkey/chaos-monkey-alternatives/
- MITRE presentation on the topic (available upon request): "Towards real time and continuous evaluation in the Cloud"

Notes

1 Yury Izrailevsky and Ariel Tseitlin, “The Next Simian Army,” July 2011.
Available: https://netflixtechblog.com/the-netflix-simian-army-16e57fbab116

2 C. Aaron Cois, “DevOps Case Study: Netflix and the Chaos Monkey,” April 30, 2015.
Available: https://insights.sei.cmu.edu/devops/2015/04/devops-case-study-netflix-and-the-chaos-monkey.html

3 Lily Hay Newman, “Friday’s Massive Comcast Outage Shows How Fragile the Internet Is,” *Wired*, June 2018.
Available: https://www.wired.com/story/friday-comcast-outage-cut-fiber

4 Arian Hornsby, “Chaos Engineering–Part 1, The Art of Breaking Things Purposefully,” June 2019.
Available: https://medium.com/the-cloud-architect/chaos-engineering--ab0cc9fbd12a

5 Gremlin, Inc., “Chaos Monkey Guide for Engineers,” 2021. Available: https://www.gremlin.com/chaos-monkey/

6 Yury Izrailevsky and Ariel Tseitlin, “The Next Simian Army,” July 2011.
Available: https://netflixtechblog.com/the-netflix-simian-army-16e57fbab116

7 C. Aaron Cois, “DevOps Case Study: Netflix and the Chaos Monkey,” April 30, 2015.
Available: https://insights.sei.cmu.edu/devops/2015/04/devops-case-study-netflix-and-the-chaos-monkey.html

8 Gremlin, Inc., “Chaos Monkey Guide for Engineers,” 2021. Available: https://www.gremlin.com/chaos-monkey/

9 National Security Agency, Cybersecurity Information, “Mitigating Cloud Vulnerabilities,” January 22, 2020. Available:
https://media.defense.gov/2020/Jan/22/2002237484/-1/-1/0/CSI-MITIGATING-CLOUD-VULNERABILITES_20200121.PDF

10 National Security Agency, “Mitigating Cloud Vulnerabilities,” Misunderstanding or misapplying “the technical controls implemented in software that define how cloud services may interact [resulting in] denial of service susceptibility or account compromise.”

11 National Security Agency, “Mitigating Cloud Vulnerabilities,” Using “weak authentication/authorization methods or include[ing] vulnerabilities that bypass these methods…[which] can allow an attacker to elevate privileges, resulting in the compromise of cloud resources.”

12 David C. Wyld, “Moving to the Cloud: An Introduction to Cloud Computing in Government,” IBM Center for the Business of Government, 2009. Available: https://www.businessofgovernment.org/sites/default/files/CloudComputingReport.pdf page 36

13 Haley Tucker, Lorin Hochstein, Nora Jones, Ali Basiri, Casey Rosenthal, “The Business Case for Chaos Engineering,” May/June 2018. Available: https://ieeexplore.ieee.org/abstract/document/8383672

14 Federal News Network, Insights by Belcan, “Test and Evaluation–What Is It and Why Do I Care,” September 11, 2018.
Available: https://federalnewsnetwork.com/federal-insights/2018/09/test-and-evaluation-what-is-it-and-why-do-i-care/

15 Alexander Kochovski, “Ransomware Statistics, Trends and Facts for 2020 and Beyond,” May 6, 2021.
Available: https://www.cloudwards.net/ransomware-statistics/

16 C. Aaron Cois, “DevOps Case Study: Netflix and the Chaos Monkey,” April 30, 2015.
Available: https://insights.sei.cmu.edu/devops/2015/04/devops-case-study-netflix-and-the-chaos-monkey.html

17 Yury Izrailevsky and Ariel Tseitlin, “The Next Simian Army,” July 2011.
Available: https://netflixtechblog.com/the-netflix-simian-army-16e57fbab116

18 Jon Brodkin, “Netflix Attacks Own Network with ‘Chaos Monkey’–and Now You Can Too,” July 30, 2012. Available:
https://arstechnica.com/information-technology/2012/07/netflix-attacks-own-network-with-chaos-monkey-and-now-you-can-too/

19 Andre Newman, “7 Important Truths About Chaos Engineering,” November 19, 2020.
Available: https://devops.com/7-important-truths-about-chaos-engineering

20 Andrew Lerner, “The Cost of Downtime,” July 16, 2014.
Available: https://blogs.gartner.com/andrew-lerner/2014/07/16/the-cost-of-downtime/

21 Andre Newman, “7 Important Truths About Chaos Engineering,” November 19, 2020.
Available: https://devops.com/7-important-truths-about-chaos-engineering

22 Department of Defense Chief Information Officer, “DoD Enterprise DevSecOps Reference Design,” Version 1.0, August 12, 2019. Available: https://dodcio.defense.gov/Portals/0/Documents/DoD%20Enterprise%20DevSecOps%20Reference%20Design%20v1.0_Public%20Release.pdf

23 Your Dictionary.Com, "White-Hat." Accessed March 13, 2020. Available: https://www.yourdictionary.com/white-hat

24 E. Tittel and E. Follis, "How to become a white hat hacker," *Business Daily News*, June 17, 2019. Available: https://www.businessnewsdaily.com/10713-white-hat-hacker-career.html

25 K. Lerwing, "Apple hired the hackers who created the first Mac firmware virus," *Business Insider*, February 3, 2016. Available: https://www.businessinsider.com/apple-hired-the-hackers-who-created-the-first-mac-firmware-virus-2016-2

26 HackerOne, "What was it like to hack the Pentagon?," h1 blog, June 17, 2016. Available: https://www.hackerone.com/blog/hack-the-pentagon-results

27 Statista.com, "Proposed federal spending by the US government on cybersecurity for selected government agencies from FY2020 to FY2021," January 25, 2021. Available: https://www.statista.com/statistics/737504/us-fed-gov-it-cyber-security-fy-budget/

28 Gartner.com, https://www.gartner.com/document/398777695. Full document behind paywall, though article about the report is available here: https://www.gartner.com/smarterwithgartner/top-actions-from-gartner-hype-cycle-for-cloud-security-2020

29 Thorben Janssen, "OOP Concept for Beginners: What Is Abstraction?," Stackify, November 23, 2017. Available: https://stackify.com/oop-concept-abstraction/

30 A firewall is a "network security device that monitors incoming and outgoing network traffic and decides whether to allow or block specific traffic based on a defined set of security rules." Cisco.Com, "What Is a Firewall?" Available: https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html

31 Internal clouds refer to "computing applications, platforms and infrastructure being delivered as a bundle of services to users within a private enterprise." Failover is "the constant capability to automatically and seamlessly switch to a highly reliable backup." References come from https://apprenda.com/library/cloud/internal-cloud-computing/ and https://www.techopedia.com/definition/1202/failover respectively.

32 "Towards real time and continuous evaluation in the Cloud." MITRE, unpublished.

33 US Department of Veterans Affairs, VA Technical Reference Model v 21.4, "Chaos Monkey." Available: https://www.oit.va.gov/Services/TRM/ToolPage.aspx?tid=11555#

34 Julia Cation, "Flight Control Breakthrough Could Lead to Safer Air Travel: The Grainger College of Engineering," March 19, 2015. Available: https://grainger.illinois.edu/news/stories/28479

35 "Latency refers to time interval or delay when a system component is waiting for another system component to do something." Available: https://techopedia.com/definition/2228/latency

36 "Server errors are typically transient and are often the result of server timeouts, throttling, or capacity limitations." Available: https://docs.aws.amazon.com/cloudsearch/latest/developerguide/error-handling.html

37 "The amount of time it takes for Failover to successfully complete." Available: https://tools.ietf.org/html/rfc6414#page-27

38 "Auto scaling is a cloud computing feature that allows users to automatically scale cloud services, like virtual machines (VM) and server capacities, up or down, depending on defined situations…Auto scaling also ensure that new instances are seamlessly increased during demand spikes and decreased during demand drops, enabling a consistent performance for lower costs." https://www.techopedia.com/definition/29432/auto-scaling

39 "Towards real time and continuous evaluation in the Cloud." MITRE, unpublished.

40 Adrian Hornsby, "Chaos Engineering–Part 1, The Art of Breaking Things Purposefully," June 2019. Available: https://medium.com/the-cloud-architect/chaos-engineering-ab0cc9fbd12a

41 Gartner.com, https://www.gartner.com/document/398777695. Full document behind paywall, though article about the report is available here: https://www.gartner.com/smarterwithgartner/top-actions-from-gartner-hype-cycle-for-cloud-security-2020

42 AWS Fault Injection Simulator. Available: https://aws.amazon.com/fis/

Authors

Jonathan Rotner is a human-centered technologist who helps program managers, algorithm developers, and operators appreciate technology's impact on human behavior. He works to increase communication and trust when working with automated processes.

Matt Yetto is an interdisciplinary systems engineer focused on emerging technologies. He works to establish new capabilities by bridging existing processes with cutting edge innovations.

The authors would like to thank Shane Smith for his contributions, and Cheryl Clopper, Casey Creech, Chris Growney, Bruce Simmons, and Dr. Amir Stephenson, for their review.

Intelligence After Next

MITRE strives to stimulate thought, dialogue, and action for national security leaders developing the plans, policy, and programs to guide the nation. This series of original papers is focused on the issues, policies, capabilities, and concerns of the Intelligence Community's analytical workforce as it prepares for the future. Our intent is to share our unique insights and perspectives surrounding a significant national security concern, a persistent or emerging threat, or to detail the integrated solutions and enabling technologies needed to ensure the success of the IC's analytical community in the post-COVID-19 world.

About MITRE

MITRE | Center for Data-Driven Policy