The Emergence of Security Chaos Engineering in DevOps

Daniel Ericsson (dericsso@kth.se)

May 5, 2023

1 Introduction

In the modern era, cloud-native applications have become the standard for software development. However, as systems are becoming increasingly more complex, and applications rely on microservices more often than ever, the threat surface of modern applications has widely increased. Companies today face significant challenges in ensuring the security of their applications and services. Traditionally, security measures have focused on attack prevention, but recently, a new proactive approach has emerged, which builds upon the principles of Chaos Engineering.

In this essay, I will give a brief overview of the origins of Chaos Engineering, before delving deeper into its applications in security settings in the form of Security Chaos Engineering. I will explore the need for this approach in today's complex and large-scale systems, list some state-of-the-art applications of Security Chaos Engineering, and reflect on its potential usage in the future.

I certify that generative AI, incl. ChatGPT, has not been used to write this essay. Using generative AI without permission is considered academic misconduct.

2 History of Chaos Engineering

2.1 Chaos Engineering

The term Chaos Engineering was coined by Netflix, its history dating back to 2008[1]. Chaos engineering is defined as "the discipline of experimenting on a system in order to build confidence in the system's capability to withstand turbulent conditions in production". Initially, Netflix ran relational databases in their own data centres, following a monolithic architecture. However, in 2008, the company migrated their services into the cloud following a major disaster in which a data centre failure shut down their entire service for three days. To avoid future incidents of the same nature, the company rebuilt their infrastructure using Amazon Web Services (AWS), breaking the entire service into multiple microservices[10]. The shift to a more complex cloud architecture introduced an increased need for resiliency, ensuring that their systems quickly recovered in the event of a failure in any of the individual nodes composing the service.

2.2 Chaos Monkey

In 2010, Netflix invented Chaos Monkey[10], a tool which became synonymous with the term Chaos Engineering for many years to come. The name came from the idea of unleashing an armed monkey into your data centre to randomly shoot down instances and chew through cables. The tool works by intentionally disabling production instances in the data centre at random, testing the resilience of the infrastructure towards the failure of individual nodes. Though a simple approach, it proved effective in exposing vulnerabilities in systems, allowing developers to detect weak points and fix these to increase the resiliency of services.

3 Landscape of Cloud-native Security

3.1 The cloud-native attack surface

The prevalence of microservices has significantly enlarged the threat surface of modern applications[11]. Applications built around a monolithic architecture could secure services by internalizing them, but this approach is no longer possible in the cloud environment due to the dependence on external components and microservices. Cloud-native applications rely on complex networks of permissions, which provide hackers with numerous attack surfaces to exploit to perform malicious attacks. The continuous deployment model has made it increasingly difficult to maintain a consistent security posture[9]. With the increased number of interconnected parts, errors made by developers can cause more damage than ever before. Accidentally exposing

keys, creating security loopholes or failing to patch known vulnerabilities can render security models put in place entirely useless.

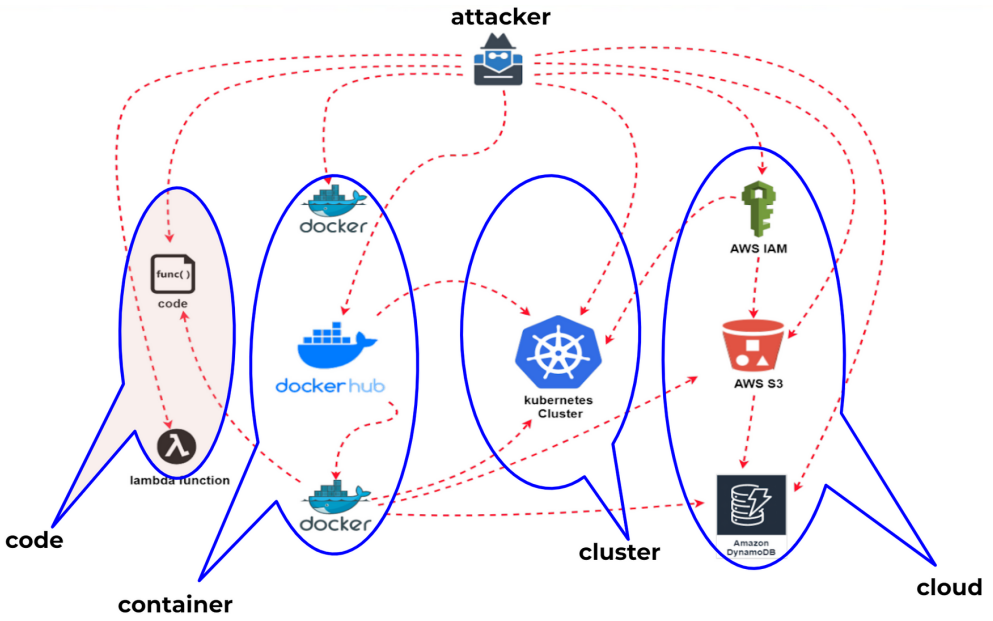

Figure 1: Visualization of the cloud-native attack surface, showing attack paths across several abstraction layers of an application[9]

3.2 The cloud-native security model

The cloud operating model typically builds on four layers of abstractions, as outlined in Figure 1: Code, container, cluster and cloud. The cloud-native security model builds upon these layers to achieve a "Defense-in-Depth" model, in what is known as the 4Cs of cloud-native security[7]. Security controls in the cloud layer are typically provided by the cloud service provider. Cluster security is the responsibility of the cluster operator, while container and code security is the responsibility of application developers.

The 4C model has various benefits and protects against a wide range of attacks, in particular those against single layers. Its weakness lies in addressing multi-layered attacks, which exploit vulnerabilities across multiple layers. Since security controls at the various abstraction layers usually operate independently, attacks that transpire across multiple layers can be difficult to detect[9].

3.3 Most common data breaches

According to the 2017 Ponemon Cost of a Data Breach Study[5], the root causes of data breaches fall into three areas, which comprise the following percentage of all incidents in total:

Malicious or criminal attacks (47%)

Human factors or errors (28%)

System glitches (25%)

Historically, cybersecurity measures have focused heavily on the prevention of malicious attacks, though as of the aforementioned report, system glitches and human errors comprise a majority of all breaches. For modern companies, a significant portion of security failures arise from poor control placement, technical misconfigurations, glitches, inadequate testing and human error. Therefore, there is an increased need to shift resources, as highlighted by prominent cybersecurity figures such as Dino Dai Zovi, Head of Security at Cash App[9], from focusing on attack prevention, to attack detection, recovery, and resistance. There is an increased need for efficient tools in place for testing security controls to ensure that vulnerabilities are identified and addressed, and security testing needs to be an integral part of the software development process.

4 History of Security Chaos Engineering

4.1 Traditional security testing

In software development, security testing has traditionally been a discrete phase in a relatively linear and static process[8], and activities such as code reviews occur late in the development process. In larger enterprises, approaches such as penetration testing and red team exercises occur long after deployment and do not integrate with the development process. With the shift to cloud-native environments and continuous deployment, the traditional approach has become insufficient for securing systems.

4.2 Security Chaos Engineering

For security testing to become a continuous practice, security professionals have begun advocating the application of Chaos Engineering to security operations. In 2018, the term Security Chaos Engineering was introduced in an article by Aaron Rinehart and Charles Nwatu[12], defined as "the discipline of instrumentation, identification,

and remediation of failure within security controls through proactive experimentation to build confidence in the system's ability to defend against malicious conditions in production". According to the authors, Security Chaos Engineering represents a paradigm shift for cybersecurity; a shift from the traditional reactive approach to security to a more proactive and continuous approach, in response to the evolving nature of modern application design.

4.3 Security Chaos Experiments

Within the paradigm of Security Chaos Engineering, so-called Chaos Experiments are conducted[8], with the goal of assessing and increasing observability within a system, to allow security teams to reduce the "unknown unknowns". Experiments intentionally introduce a failure mode which allows teams to assess how observable and measurable their security systems are, and identify weaknesses. These experiments differ greatly from traditional security testing, as they do not introduce external events, such as with a simulated attack, and only involve making a single change to a system, in order to observe what happens. The approach is mostly evidence-based, in contrast to traditional approaches, which rely on assumed security postures[3].

The process of a Chaos Experiment typically involves four phases, which are outlined below.

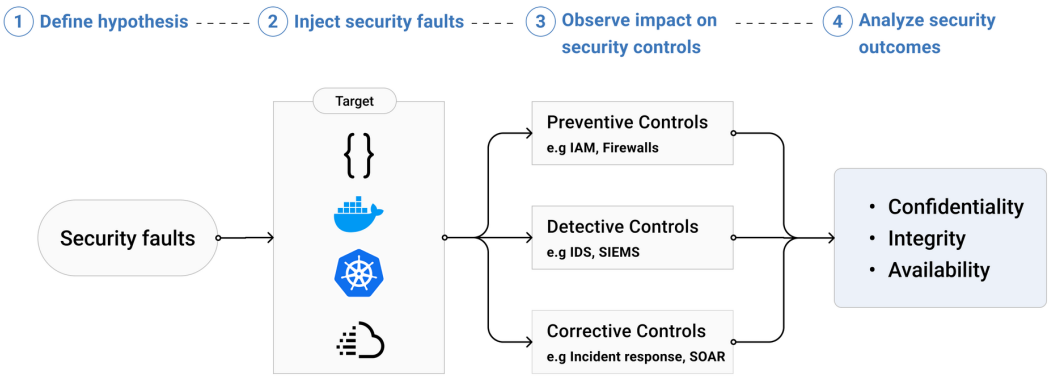

Figure 2: An overview of the steps of conducting a Chaos Experiment[9]

Define hypothesis

The first step in a Chaos Experiment is to define a hypothesis for an action or event

which could cause a security fault.

Inject security faults
A hypothesized security fault is injected into a target system. The target could be anything from a Lambda function to an entire cloud infrastructure.

Observe impact on security controls
After the security fault is injected, the impact on different security controls is carefully observed. Typically, these controls fall into three categories: detective, preventive and recovery controls.

Analyze security outcomes
Finally, the results are analyzed and the hypothesis is either validated or rejected, based on how the security controls responded to the security fault. The impact of the fault is evaluated in terms of whether any violations occurred with respect to the CIA triad, which stands for Confidentiality, Integrity, and Availability.

By injecting security faults into cloud-native applications, Chaos Experiments produce empirical data and insights into the behaviour of security controls, allowing for proactive and iterative security hardening[13]. There are currently no established guidelines for practising Security Chaos Engineering, but experimentation is at its core. For instance, Kennedy A. et al. propose using the term Risk Driven Fault Injection[14], for the approach of designing a security hypothesis, injecting security faults and analyzing the results to gain insights into security models for cloud infrastructures.

5 Current State of Security Chaos Engineering

5.1 ChaoSlingr

ChaoSlingr is one of the first tools ever written for Security Chaos Engineering[6] and was released in 2017. The tool, which focuses on experimentation on AWS infrastructure, is written in Python and contains three major Lambda functions called Slinger, Tracker, and Generator. The Generator function identifies where a Chaos Experiment can be run in a system based upon AWS security tags, and what will be affected by the fault. Slinger then injects the failure, for instance by opening a closed port and introducing a new condition to the environment, to understand how well the security response to it is. Finally, as the name implies, Tracker keeps track of what happened in terms of the events with the tool, provides logging, and can be used to report the information to Slack so that developers can respond in real-time. Although ChaoSlingr is no longer actively maintained, the project is

open-source and provides a useful framework for writing Chaos Experiments.

5.2 ChAP

Another tool released in 2017 is the Chaos Automation Platform[2], or ChAP, which is another addition to the chaos tooling family developed by Netflix. The tool was created to automate experiments due to the company's rapid change in production systems, which necessitated continuous experimentation. ChAP is integrated with the CI/CD system of Netflix and "interrogates" the deployment pipeline for a user-specified service, then launches experiments and control clusters of that service, routing a small amount of traffic to each. A specified failure is injected, and the results are reported to the service owner. It does all of this in production, as to align with the principles of Chaos Engineering. The tool automatically ends an experiment should it exceed a predefined error budget.

5.3 Verica

Verica was founded by Aaron Rinehart and Casey Rosenthal in 2019[6], the former of which helped coined the term Security Chaos Engineering, and the latter of which ran the Chaos Engineering team at Netflix. The tool provides an enterprise platform based on ChAP, called "Continuous Verification", which includes security-specific Chaos Experiments. The tool integrates with Kubernetes and Kafka out of the box and can be used in the cloud or on-premises.

5.4 CloudStrike

CloudStrike is a cloud security system that implements the principles of Chaos Engineering, proposed by Kennedy A. et al. in 2019[15]. The tool conducts Chaos Experiments on AWS and GCP, and builds a measurable hypothesis using the concept of expected state, the secure state of a cloud resource at a point in time, known by the resource orchestration system. The tool then orchestrates random actions against the target cloud system, such as the creation, deletion and modification of resources, including user and bucket configurations.

6 Future State of Security Chaos Engineering

6.1 Chaos Engineering in Machine Learning

The increasing use of machine learning has introduced new risks for modern systems which incorporate such models into their services. Machine learning workflows are dynamic and complex, often functioning as a black box[4], and difficult for humans

to analyze. Changing the input of a model slightly or introducing noise can easily lead to unexpected results. Consequently, ML models can be especially vulnerable to attacks. I believe that Chaos Engineering will play an important role in the near future, as machine learning models are more rapidly being deployed than ever. Chaos Engineering, and Chaos Experiments, allow for a proactive approach to gaining valuable insights into these models and may prove to be more effective than traditional methods.

6.2 A new paradigm shift or a passing phase?

It is clear that the idea of Security Chaos Engineering is gaining traction, and I think that the approach will be more widely used in the future. Relying on preventive measures will only get you so far, and more proactive approaches are needed as systems become increasingly more complex and are subject to constant change. A large portion of security breaches today are made possible due to human error and misconfiguration. Consequently, one of the biggest problems companies face today is not that they have inadequate security models in place, but rather that they fail to test these models thoroughly to ensure that they work as intended.

7 Conclusions

The key takeaway from this essay is that Security Chaos Engineering is a powerful proactive approach to cloud-native security that allows for valuable insights into the behaviour of security controls. As cloud-native applications are becoming increasingly complex, and the landscape is changing quickly, this proactive approach is becoming increasingly important for companies that need their systems to be secure. The approach will likely be more commonplace in the future, especially as machine learning models are being utilized more often.

References

[1] Channel Asia. *Ten years on: How Netflix completed a historic cloud migration with AWS*. https://www.channelasia.tech/article/646530/ten-years-how-netflix-completed-historic-cloud-migration-aws. Accessed: 2023-05-05.

[2] Netflix Technology Blog. *ChAP: Chaos Automation Platform*. https://netflixtechblog.com/chap-chaos-automation-platform-53e6d528371f. Accessed: 2023-05-05.

[3] Mathias Conradt and Kennedy A. Torkura. *Defeating Ransomware Attacks With Security Chaos Engineering*. Nov. 2022.

[4] Consulteer. *Chaos Engineering: Key Future Trends*. https://www.consulteer.com/chaos-engineering-key-future-trends/. Accessed: 2023-05-05.

[5] IBM. *2017 Ponemon Institute Cost of a Data Breach Study*. 2017.

[6] Kyle Johnson. *Tools to conduct security chaos engineering tests*. https://www.techtarget.com/searchsecurity/feature/Tools-to-conduct-security-chaos-engineering-tests. Accessed: 2023-05-05.

[7] Kubernetes. *Overview of Cloud Native Security*. https://kubernetes.io/docs/concepts/security/overview/. Accessed: 2023-05-05.

[8] Jamie Lewis and Chenxi Wang. *Chaos Engineering: New Approaches To Security*. Aug. 2019.

[9] Mitigant. *Demystifying Security Chaos Engineering*. https://www.mitigant.io/blog/demystifying-security-chaos-engineering-part-i. Accessed: 2023-05-05.

[10] Ales Plsek and Rob Hilton. *The evolution of chaos engineering at Netflix*. Dec. 2022.

[11] ProtectOnce. *How Your Attack Surface Changes When Moving To Cloud Native Apps*. https://protectonce.com/how-your-attack-surface-changes-when-moving-to-cloud-native-apps/. Accessed: 2023-05-05.

[12] Aaron Rinehart and Charles Nwatu. *Security Chaos Engineering: A new paradigm for cybersecurity*. https://opensource.com/article/18/1/new-paradigm-cybersecurity. Accessed: 2023-05-05. Jan. 2018.

[13] Kelly Shortridge. "From Lemons to Peaches: Improving Security ROI through Security Chaos Engineering". In: *2022 IEEE Secure Development Conference (SecDev)*. 2022, pp. 59–60. DOI: 10.1109/SecDev53368.2022.00021.

[14] Kennedy A. Torkura et al. "CloudStrike: Chaos Engineering for Security and Resiliency in Cloud Infrastructure". In: *IEEE Access* 8 (2020), pp. 123044–123060. DOI: 10.1109/ACCESS.2020.3007338.

[15] Kennedy A. Torkura et al. “Security Chaos Engineering for Cloud Services: Work In Progress”. In: *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*. 2019, pp. 1–3. DOI: 10.1109/NCA.2019.8935046.