Koney · Cyber Deception Policies for Kubernetes

The Honeynet Project Workshop 2025

June 2, 2025 · Prague, Czech Republic

PRESENTER

Mario Kahlhofer
Dynatrace Research

CO-DEVELOPER

Matteo Golinelli
University of Trento

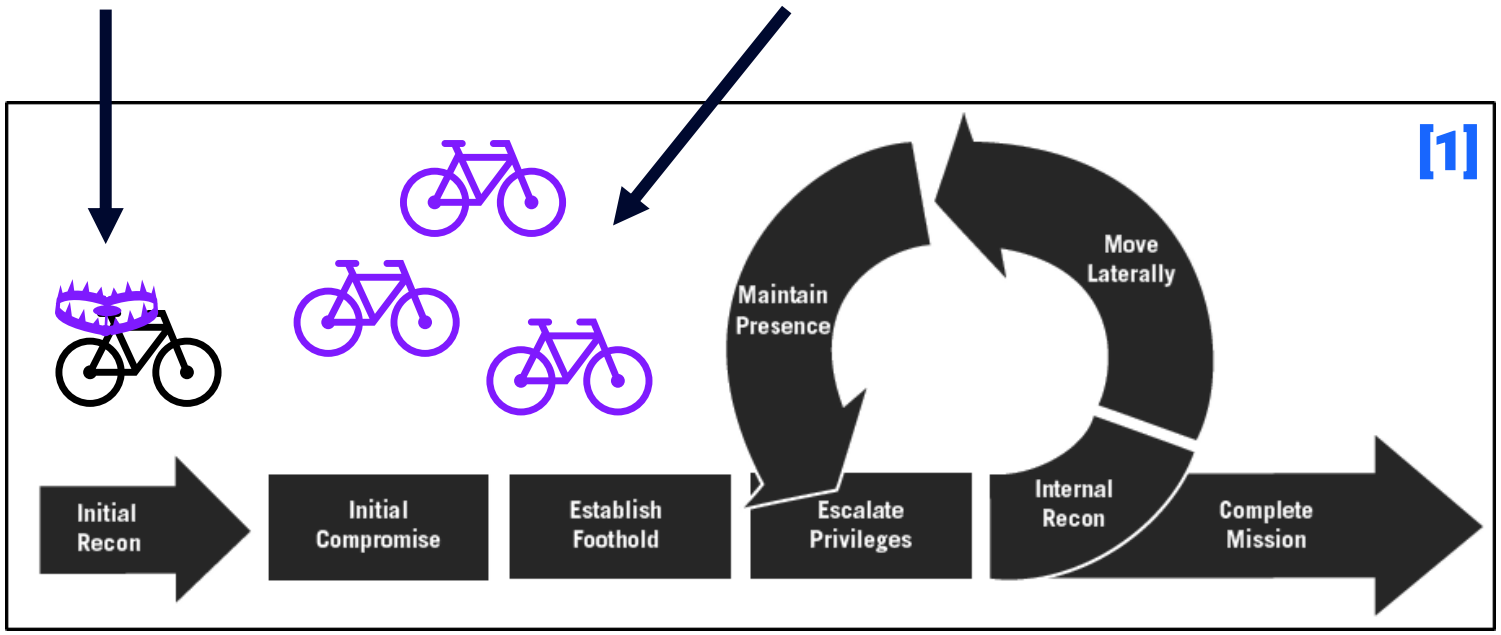
dynatrace

APPLICATION LAYER CYBER DECEPTION

Let's Embed Traps Directly Into Applications!

Thieves shall not steal your bike

Let's place some fake bikes then



'Classic' honeypots are isolated fake applications, reachable over the network

Talk Focus

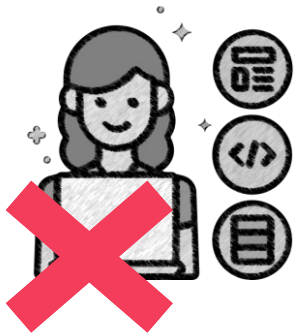
Honeytokens & application layer deception techniques are traps embedded into applications & systems

[1] Mandiant, "APT1: Exposing One of China's Cyber Espionage Units," 2013. [Online].

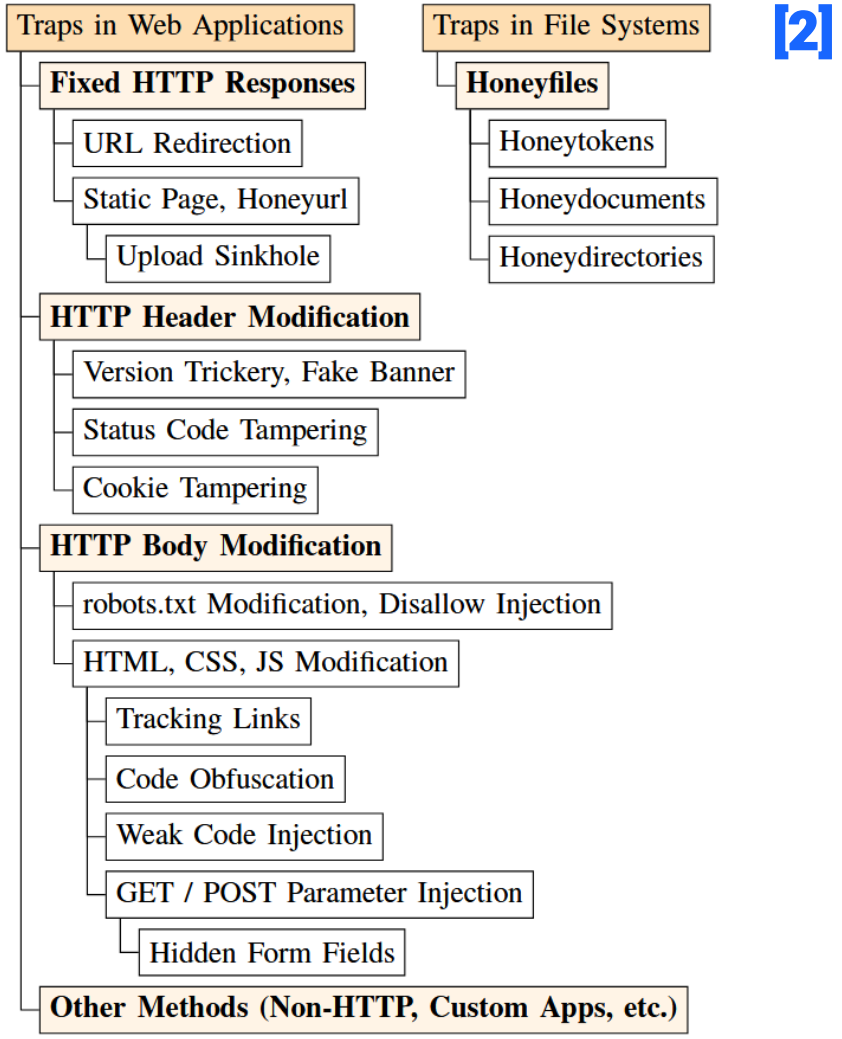
APPLICATION LAYER CYBER DECEPTION

Let's Embed Traps Directly Into Applications! (cont.)

- Place **Honeytokens** in (container) filesystems
- Add new **HTTP endpoints** to mislead hackers
- Modify **HTTP headers**, e.g., version numbers
- Modify **HTTP bodies**, e.g., hidden form fields
- Other (non-HTTP) methods



But software applications are rarely deployed by the team that wrote the code, and often the responsibility for security measures lies entirely elsewhere.



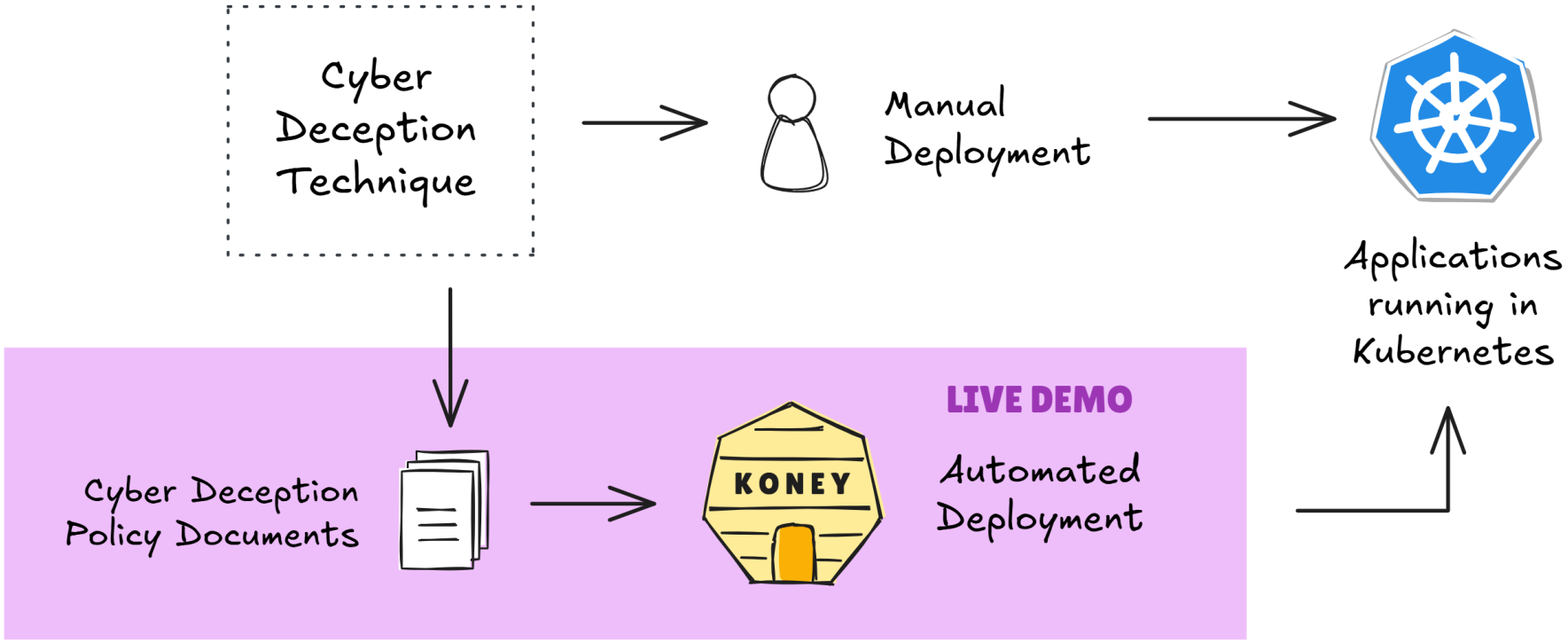
[2]

[2] M. Kahlhofer, M. Golinelli, and S. Rass, "Koney: A Cyber Deception Orchestration Framework for Kubernetes," 2025, arXiv: arXiv:2504.02431. doi: 10.48550/arXiv.2504.02431.

THE IDEA

Automated Deployment of Cyber Deception “As Code”

… instead of manually deciphering cyber deception techniques from academic papers.



THE IDEA

Cyber Deception Policy Documents

honeytoken.yaml — YAML

```yaml
apiVersion: koney.io/v1alpha1
kind: DeceptionPolicy
spec:
  traps:
    - filesystemHoneytoken:
        filePath: /run/secrets/token
        fileContent: "secret"
      match:
        any:
          - resources:
              selector:
                matchLabels:
                  op/honeytoken: true
      decoyDeployment:
        strategy: containerExec
      captorDeployment:
        strategy: tetragon
```

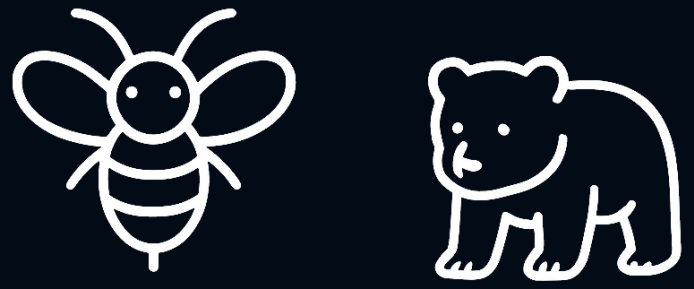
← Trap-specific parameterization

← Criteria for selecting the workloads (e.g., containers) in which to deploy the traps

← **Decoy. [3]** Strategy to deploy the trap itself

← **Captor. [3]** Strategy for monitoring the trap

[3] W. Fan, Z. Du, D. Fernández, and V. A. Villagrá, "Enabling an Anatomic View to Investigate Honeypot Systems: A Survey," IEEE Sys. J., vol. 12, no. 4, pp. 3906–3919, Dec. 2018, doi: 10.1109/jsyst.2017.2762161.

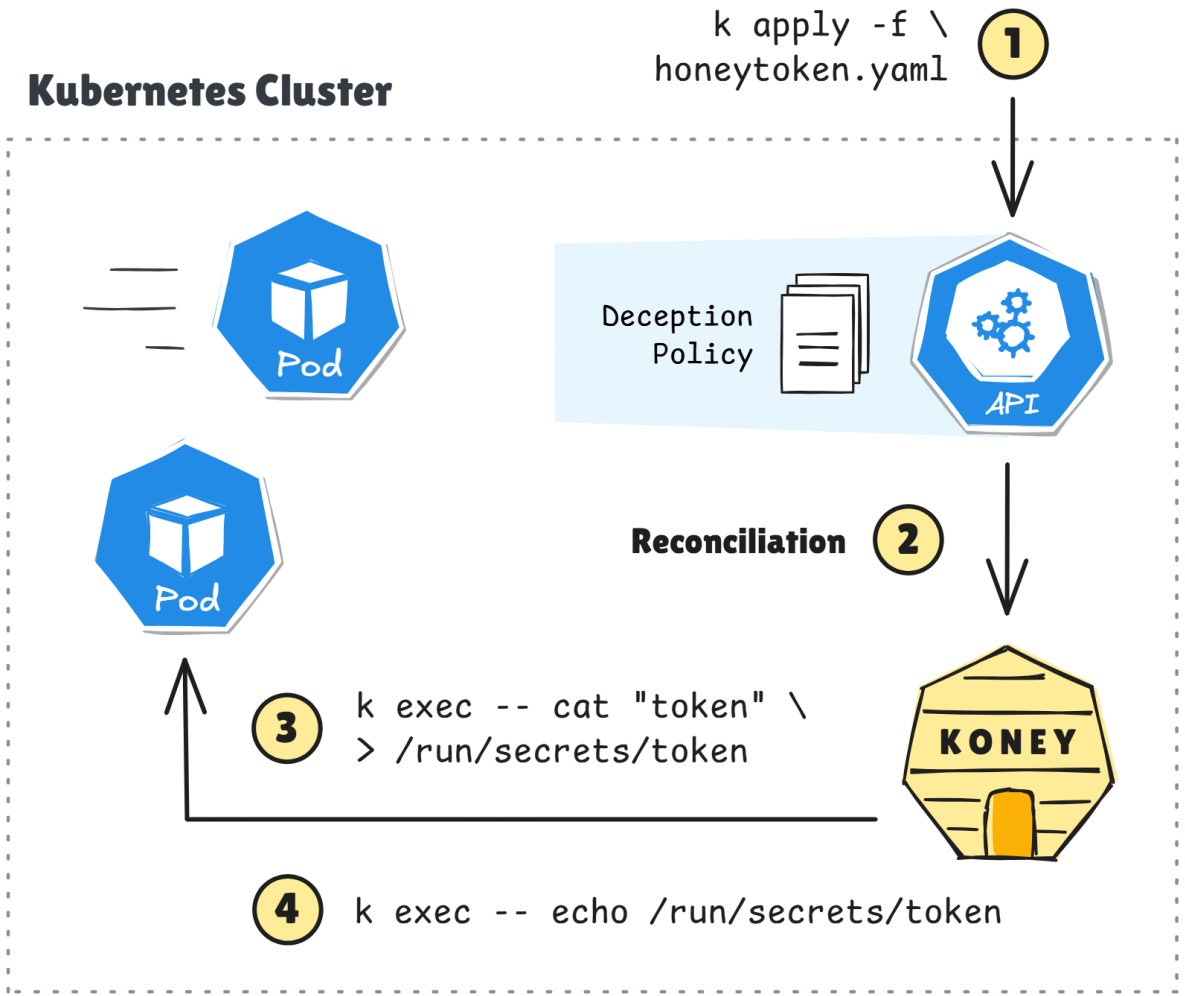


LIVE DEMO

Applying a DeceptionPolicy With Koney

DECOY STRATEGY

Placing Honeytokens by Executing Shell Commands



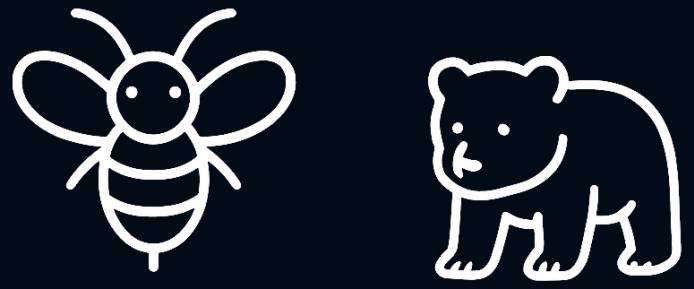
honeytoken.yaml — YAML

```yaml
apiVersion: koney.io/v1alpha1
kind: DeceptionPolicy
spec:
  traps:
    - filesystemHoneytoken:
        filePath: /run/secrets/token
        fileContent: "secret"
      match:
        any:
          - resources:
              selector:
                matchLabels:
                  op/honeytoken: true
      decoyDeployment:
        strategy: containerExec
      captorDeployment:
        strategy: tetragon
```

Placing Honeytokens by Executing Shell Commands (cont.)

Deployment	`cat "secret" > /run/secrets/token`
Verification	`echo /run/secrets/token`
Clean-Up	`rm /run/secrets/token`

Monitoring of access attempts	?
Transparency for system operators	✓ **DeceptionPolicy**
Zero Downtime of application services	✓ yes
Non-interference with genuine operation	✓ just a few process executions



LIVE DEMO

Honeytoken File Access Monitoring

LIVE DEMO

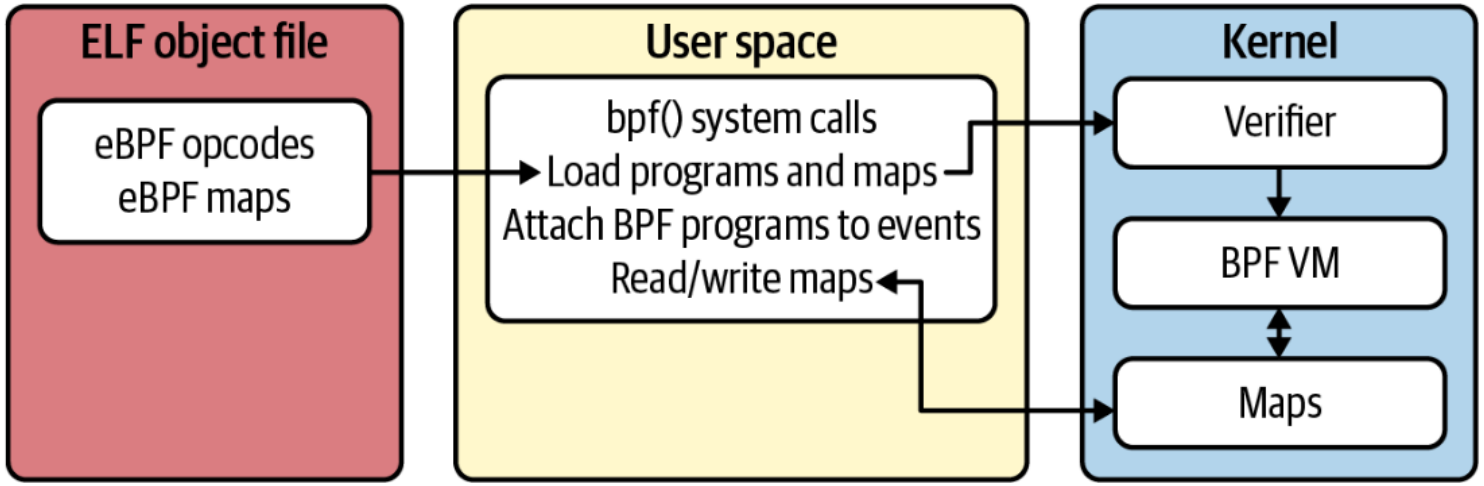
Koney Alert Example

alert.json JSON


```
{
  "timestamp": "2025-06-02T11:17:02Z",
  "deception_policy_name": "deceptionpolicy-servicetoken",
  "trap_type": "filesystem_honeytoken",
  "metadata": { "file_path": "/run/secrets/koney/service_token" },
  "pod": {
    "name": "koney-demo-deployment-5bcbd78875-45qpn",
    "namespace": "koney-demo",
    "container": {
      "id": "docker://e19c1827e255ce7a5c5fd74eb4ee861388f83a16410effd65e30d3b051cd815f",
      "name": "nginx"
    }
  },
  "process": {
    "pid": 148373, "uid": 0, "cwd": "/", "binary": "/usr/bin/cat",
    "arguments": "/run/secrets/koney/service_token"
  }
}
```

File Access Monitoring with eBPF

eBPF makes the kernel programmable. eBPF programs are typically written in a subset of C or Rust and compiled to an object file.

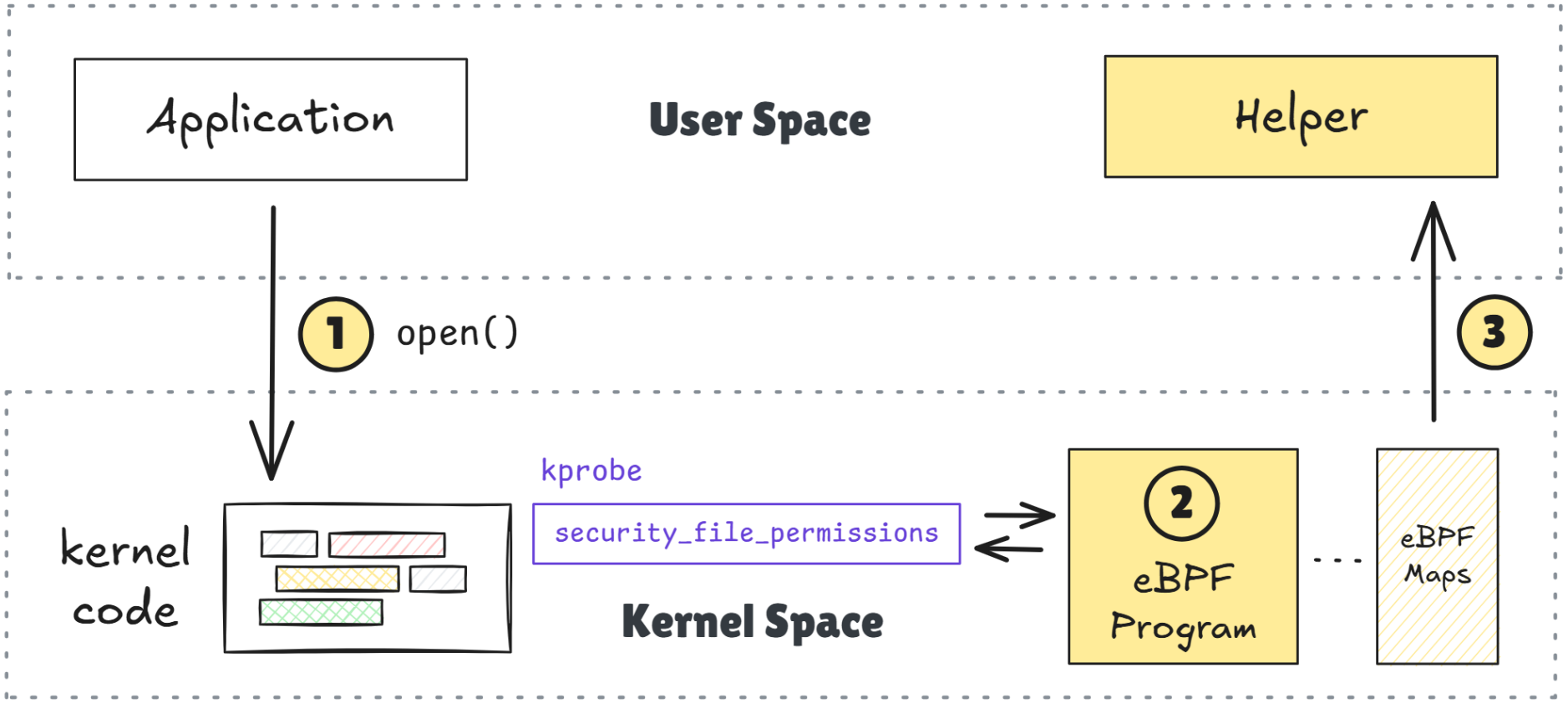


[4]

[4] L. Rice, What Is eBPF? Sebastopol, CA: O'Reilly Media, Inc., 2022. Accessed: Sep. 16, 2022. Available: https://isovalent.com/books/ebpf/

File Access Monitoring with eBPF (cont.)

We hook the **security_file_permissions** kprobe in kernel space.



File Access Monitoring with eBPF (cont.)

policy.yaml YAML

```yaml
apiVersion: cilium.io/v1alpha1
kind: TracingPolicy
metadata:
  name: monitor-honeytoken
spec:
  kprobes:
    - call: security_file_permission
      syscall: false
      return: true
      args:
        - index: 0
          type: file
        - index: 1
          type: int
      returnArg:
        index: 0
        type: int
      returnArgAction: Post
      selectors:
        - matchArgs:
            - index: 0
              operator: Prefix
              values:
                - /run/secrets/token
```

[5]

Tetragon is a Kubernetes Operator that simplifies the creation of "tracing policies" in K8s clusters.

Falco and **Tracee** are popular alternatives.

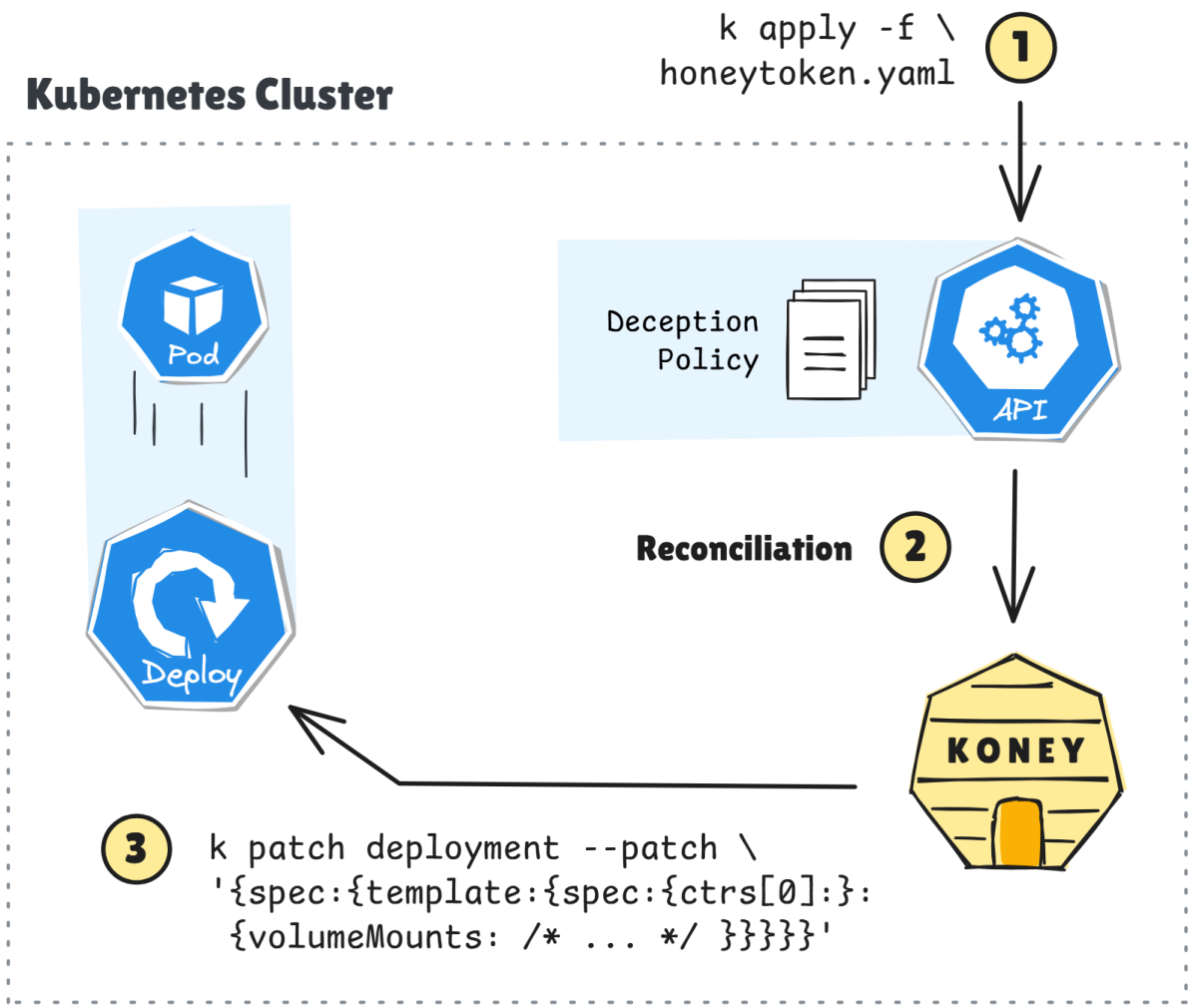
[5] K. Kourtis and A. Papagiannis, "File Monitoring with eBPF and Tetragon (Part 1)," Isovalent Blog. Accessed: Feb. 2025. [Online]. Available: https://isovalent.com/blog/post/file-monitoring-with-ebpf-and-tetragon-part-1/

LIVE DEMO

Applying a DeceptionPolicy With Koney to a Distroless Container Image

DECOY STRATEGY

Placing Honeytokens by Mounting Volumes



deployment.yaml — YAML

```yaml
apiVersion: apps/v1
kind: Deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - image: nginx:latest
          name: web
          volumeMounts:
            - mountPath: /run/secrets/token
              name: koney-volume
              readOnly: true
      volumes:
        - name: koney-volume
          secret:
            secretName: koney-secret
```

DECOY STRATEGY · SUMMARY

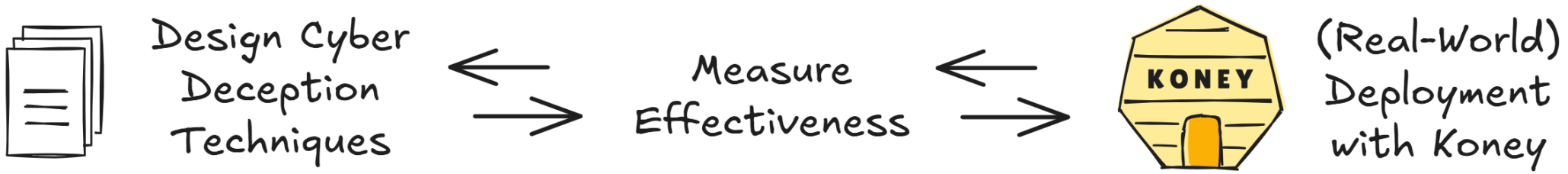
Placing Honeytokens With Koney

	Strategy 1: Shell Commands	Strategy 2: Volume Mounts
Deployment	`cat "secret" > /run/secrets/token`	`+spec.containers.volumeMounts`
Verification	`echo /run/secrets/token`	`echo /run/secrets/token`
Clean-Up	`rm /run/secrets/token`	`-spec.containers.volumeMounts`
Monitoring of access attempts	✓ **eBPF (via Tetragon)**	✓ **eBPF (via Tetragon)**
Transparency for system operators	✓ **DeceptionPolicy**	✓ even better, visible manifest change
Zero Downtime of application services	✓ yes	✗ **needs container restart in Kubernetes**
Non-interference with genuine operation	✓ just a few process executions	✓ even better, no process executions

OUTLOOK

Deceive. Test. Repeat.

We expect to speed up the cycle time between cyber deception design and deployment, and to help separate the responsibilities of **deception technology authors**, **software application developers**, and **system operators**.



Development Outlook

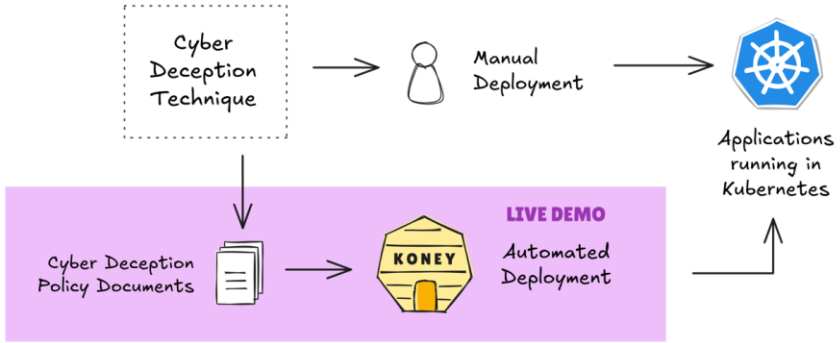
- Traps for HTTP-based applications
- eBPF Monitoring without Tetragon
- LLM-generated policy documents

github.com/dynatrace-oss/koney

THE IDEA

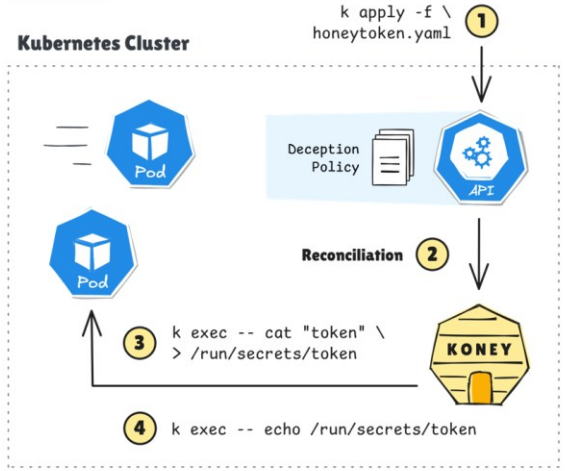
Automated Deployment of Cyber Deception “As Code”

… instead of manually deciphering cyber deception techniques from academic papers.



DECOY STRATEGY

Placing Honeytokens by Executing Shell Commands



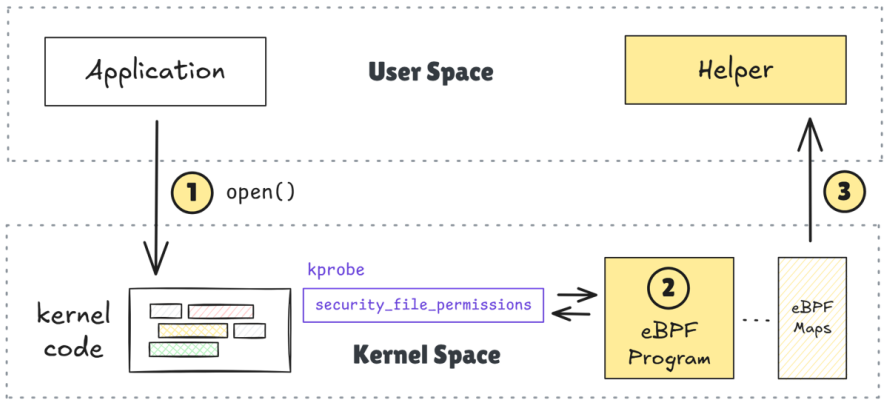
honeytoken.yaml YAML

```yaml
apiVersion: koney.io/v1alpha1
kind: DeceptionPolicy
spec:
  traps:
    - filesystemHoneytoken:
        filePath: /run/secrets/token
        fileContent: "secret"
      match:
        any:
          - resources:
              selector:
                matchLabels:
                  op/honeytoken: true
      decoyDeployment:
        strategy: containerExec
      captorDeployment:
        strategy: tetragon
```


CAPTOR STRATEGY

File Access Monitoring with eBPF (cont.)

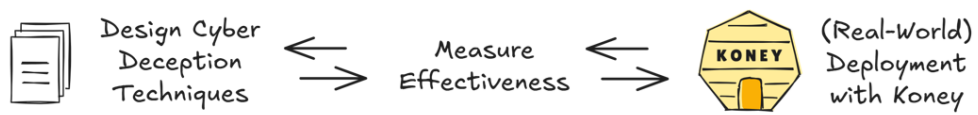
We hook the **security_file_permissions** kprobe in kernel space.



OUTLOOK

Deceive. Test. Repeat.

We expect to speed up the cycle time between cyber deception design and deployment, and to help separate the responsibilities of **deception technology authors**, **software application developers**, and **system operators**.



Development Outlook

- Traps for HTTP-based applications
- eBPF Monitoring without Tetragon
- LLM-generated policy documents

github.com/dynatrace-oss/koney