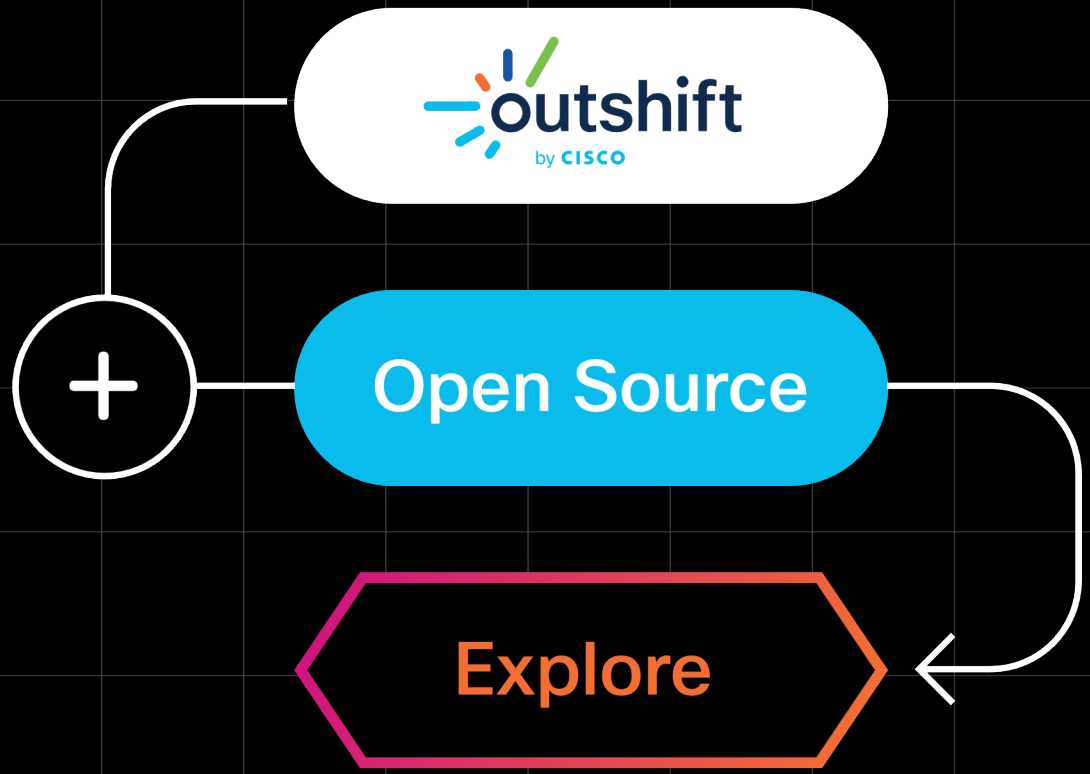
outshift by CISCO

The Evolution of Zero-Trust with Workload-Level Identity



Nándor Krácser
Engineering Lead

Nandor is an experienced software expert, specializing in security and network engineering. At Banzai Cloud, Nandor's efforts were key in advancing many open-source projects, most notably the acclaimed Bank-Vaults project, which simplified the secret management of Kubernetes-based applications. Following Banzai's integration into Cisco, Nandor is now focused on crafting a zero-trust solution, incorporating extensive WebAssembly elements, designed to secure applications operating on any level above the Linux kernel.

Márton Sereg
Senior Product Manager

As a PM, Marton is working on a Kernel space solution for zero trust networking. With a robust software engineering background, he formerly contributed to Outshift’s R&D, primarily focusing on service meshes and server-side WebAssembly. A passionate advocate for open source, Marton's journey before joining Cisco includes pivotal work at a startup, where he was instrumental in launching several notable open source projects in the cloud-native space. Among these, "logging-operator" and "bank-vaults" stand out, both on track to become CNCF sandbox projects.

Lateral movements in practice

A simplified example of how a security breach happens

outshift by cisco

Attack narrative

1. Phishing attack - AWS credentials are stolen
2. The attacker starts a VM inside a security group
3. Scanning of the internal network
4. Exploiting trust relations – access user management API
5. Data Exfiltration – retrieving internal user info

Threats are evolving

7 months before breaches are identified and contained

60% of security breaches contain lateral movements

4% of alerts are even investigated properly

What could we do to prevent the breach?

1.	Employee awareness against phishing	Everyone can be a victim...
2.	Principle of least privilege for credentials	Someone will eventually have access...
3.	Do sophisticated network segmentation	It is just limiting the scope...
4.	Set up fine-grained network policies	Can be extremely complex...
5.	Use authentication for internal services	It’s just hard...
6.	Active monitoring of anomalies	And it’s even harder...

Would a Zero Trust strategy solve everything?

And really, what do we mean by Zero Trust?

outshift by CISCO

What is Zero Trust?

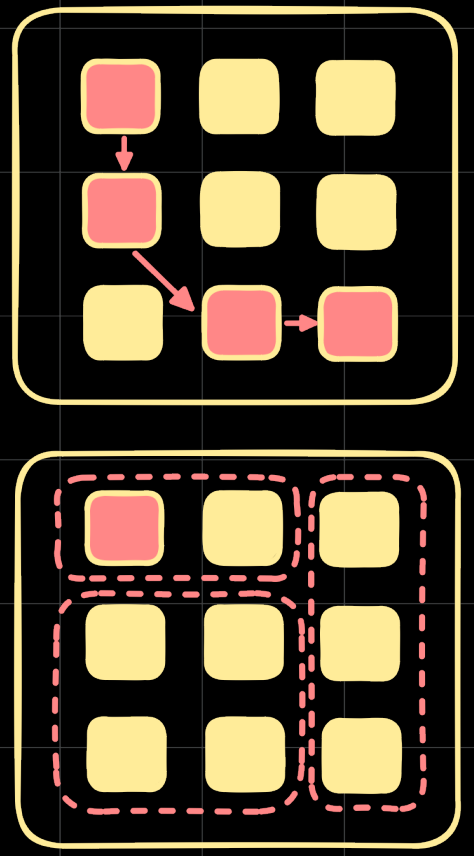
- It’s a security principle or a strategy, not a specification, an implementation, or even one product
- Moving from a ‘trust but verify’ to a ‘never trust, always verify’ model
- Driven by increasingly distributed and complex application- and data architectures
  - Perimeter security is becoming obsolete
  - Rise in attacks involving lateral movement
  - Need for granular workload access control

Areas of Zero Trust

1. Zero Trust Network Access
    - Secure remote access to an organization’s applications, services, and data
    - Based on clearly defined access control policies, instead of granting access to the whole network, like a VPN
2. ~~Microsegmentation~~ - Workload-to-workload Zero Trust
    - Granular access controls that are closer to the workload
    - Abstracting the firewall function

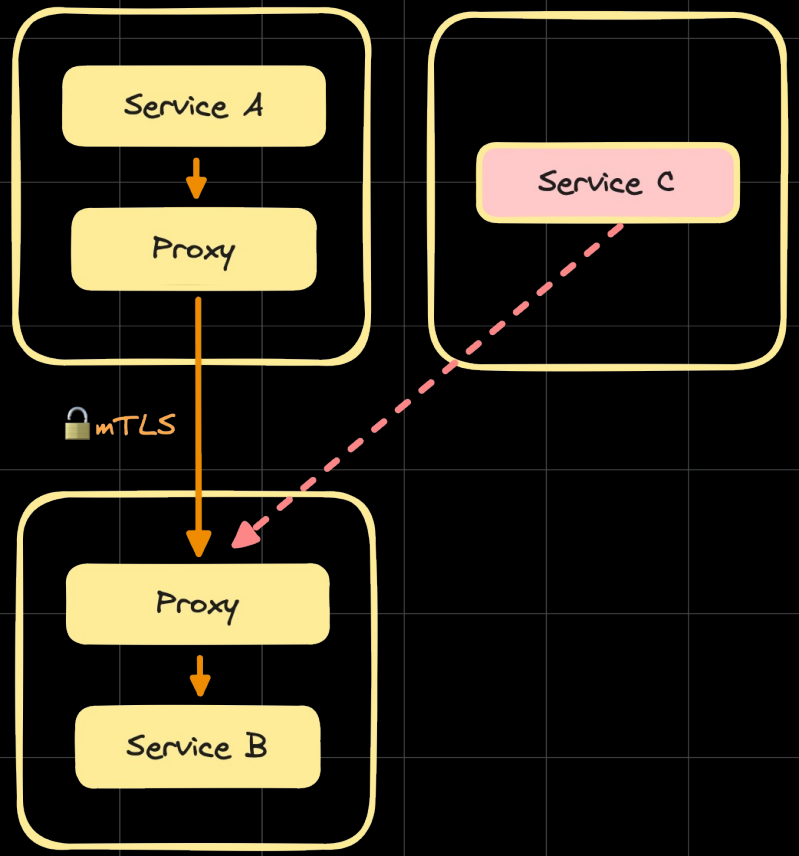
1: Microsegmentation

1. Dividing a network into segments and applying security controls to each
2. Network-based, reduce attack surface
3. Doesn’t encrypt traffic
4. Doesn’t work well in Kubernetes
5. Complexity in setting up policies in a changing environment



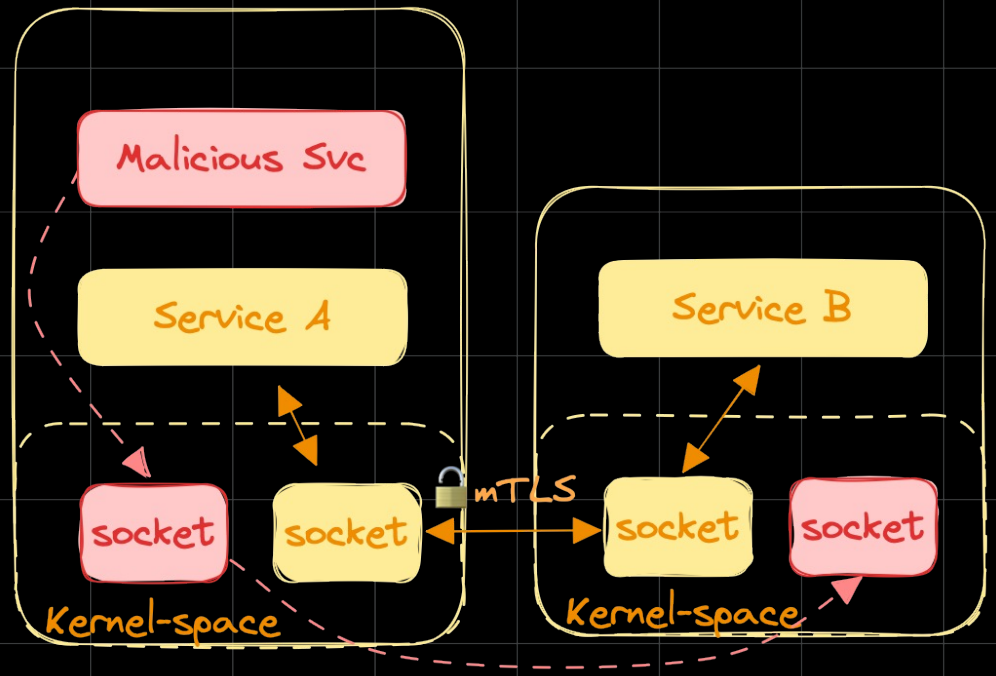
2: Service meshes

1. Only work realistically on Kubernetes
2. Mixes responsibilities between network and security teams
3. Inherently trust everything running in the pod behind the sidecar
4. Comes with the proxy hell



3: Kernel-level Identity & Encryption

1. Bind identities to processes and solve mTLS in Kernel-space
2. Works everywhere with a Linux host
3. Works natively with Kubernetes through a connector
4. Access control and also encryption
5. Application and network agnostic

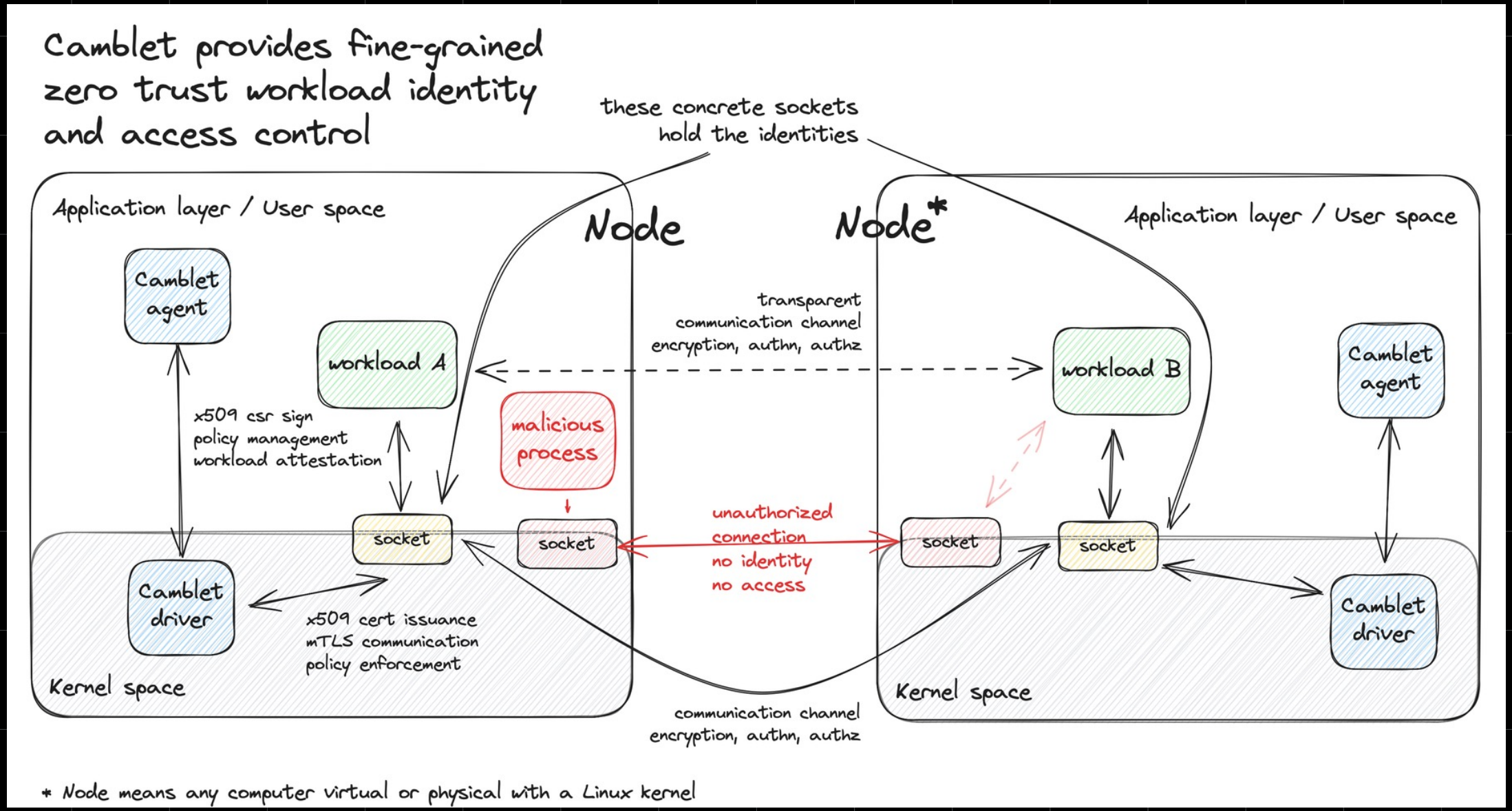


This is Camblet!

A new, open source project for automatic Kernel-space workload identity, access control, and encryption

outshift by CISCO

How does Camblet work?



The core of Camblet is in the Kernel

- The complete TLS handshake happens in Kernel space
- It’s standard TLS – compatible with other sources or destinations
- Unencrypted traffic never leaves Kernel space
- Private keys don’t need to leave Kernel space
- It gives us socket - level identity – unauthorized processes can’t have access even if they run in the same container…

But we still need a user space component

- A Camblet agent needs to run on all nodes
- Certificate signing happens through this agent
- Metadata enrichment - the process of collecting metadata of workloads in different environments like containers or orchestration systems
- But Camblet is fully transparent to users (no rebuild, no restart)

Identity & access policies

- Camblet uses a simple policy configuration based on identities
- Identities are described as SPIFFE IDs
- SPIFFE IDs are present in certificates
- Identities are defined through metadata selectors
- Metadata can include environment-specific elements (like K8s labels)

Service discovery

- On the kernel level Camblet only knows IP addresses and ports
- Service discovery “provides DNS in Kernel space”
- It defines which workloads are part of the system
  - But it isn’t needed for SPIFFE IDs
- Currently, a user’s responsibility to describe the system
- Automatic connectors to existing service discovery solutions are planned

How do I use Camblet?

Install:

- To all nodes where Camblet should work
- Installer provisions both agent and kernel modules
- `curl -L camblet.io/install.sh | bash`

After install:

- Write and distribute security policies
- Write and distribute service discovery files

Demo time...

Learn more

Start at camblet.io

Github: https://github.com/cisco-open/camblet

outshift by cisco

Questions?