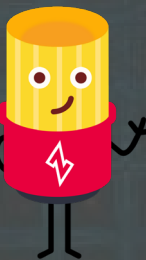


Secure Software by Design 2024

PROTECTING APIS WITH ZERO TRUST OVERLAY MESH NETWORKS



CLINT DOVHOLUK
 **NETFOUNDRY**

ABOUT ME



THIS IS DEFINITELY NOT ME



Developers don't **care**
about security

VERY COMMON THEME



do developers care about security



ITPro Today

<https://www.itprotoday.com/devsecops/how-get-de...>

How to Get Developers to Go All-In on Security

Apr 22, 2022 — Software Developers and Security ... **developers don't care**
Most developers do take security at least pretty seriously.



TechRepublic

<https://www.techrepublic.com/article/developers-d...>

Developers do not view application security

Apr 6, 2022 — According to Secure Code Warrior's State of DevSecOps



Invicti

<https://www.invicti.com/blog/web-security/develop...>

So your developers don't care about security? They ...

Mar 23, 2023 — Even with extensive developer education and the best intentions of the developers, **security will not be a priority if management does not** ...



TechBeacon

<https://techbeacon.com/security/why-developers-di...>

Why developers dislike security—and what you can do ...

Developers dislike security but won't always admit it. They dislike the security function because



Australian Computer Society

<https://ia.acs.org.au/article/why-developers-don-t-p...>

Why developers don't prioritise security

Oct 20, 2022 — While the majority of developers say they are willing to champion security and commit to higher standards of code **quality, they can't do** ...

<https://www.reddit.com/cybersecurity/comments/do...>

do developers care about security? : r/cybersecurity

2022 — **No.** Developers are rarely tasked with delivering a secure product. They are asked to deliver software quickly. The old rule is security applies ...

50 answers · Top answer: Some of us do, but it's difficult to find security training for our perspe...



“SHIPPING IS A FEATURE”



ZERO TRUST
IN MY APP?



A man with a beard and long hair, wearing a brown coat, is shown in a post-apocalyptic setting. He is holding a handgun in his right hand and looking back over his shoulder. The background is filled with debris and ruins of buildings.

ZERO TRUST
IN MY APP...

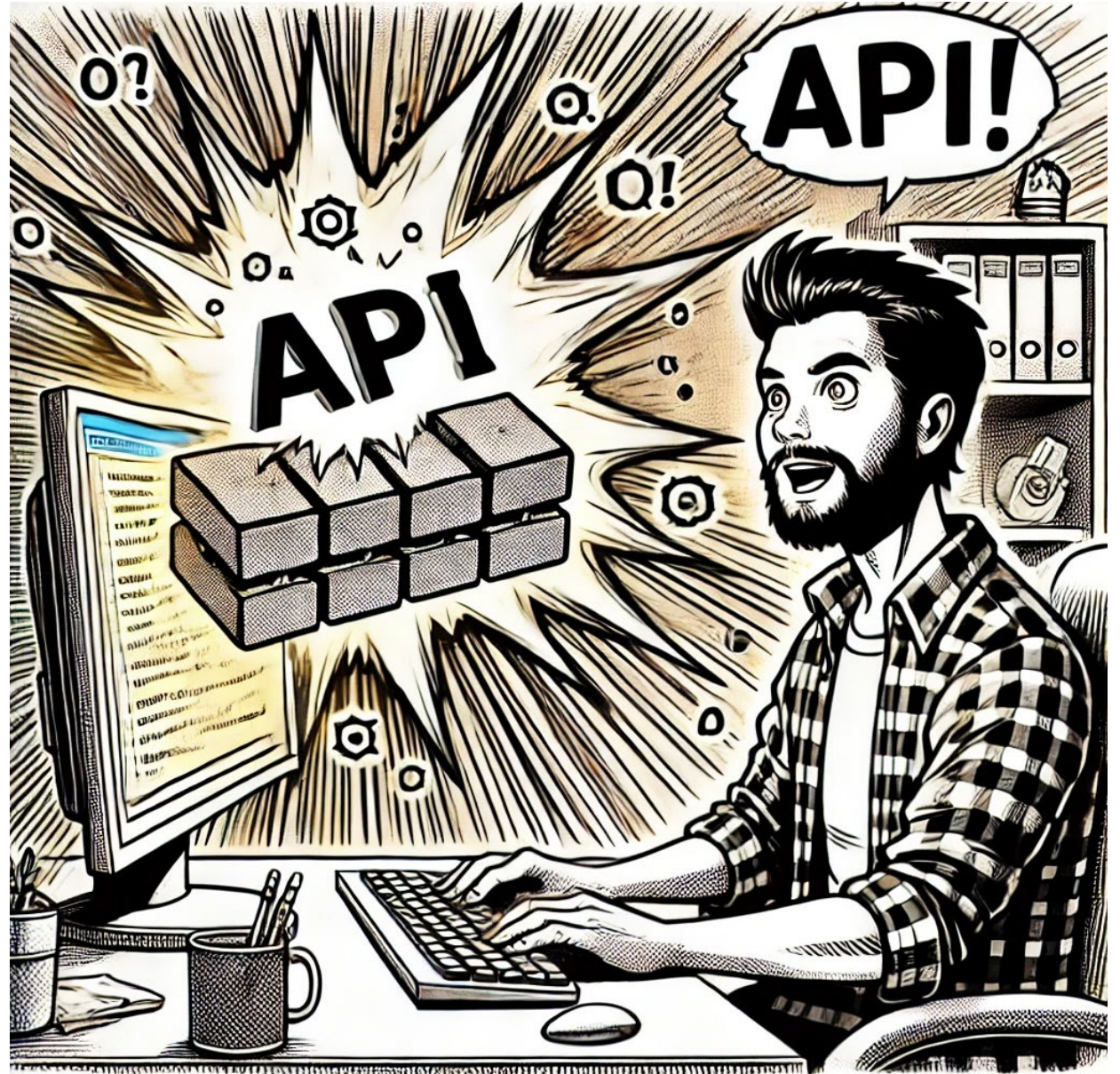
LET'S SECURE SOME APIS

- Long-time developer - 20+ years
- Grew up with Java -> C# -> Golang -> C
- Mostly API work
- Telecom -> IoT -> Zero Trust Networking



(THIS IS ME)

What is a
secure by design
API?





“The Beast”

vs

“Mad Max”

Zero Trust

The diagram features a central circular area with a dark gray background, overlaid with a faint, complex network of white nodes and lines. Surrounding this central area are three teardrop-shaped segments, each with a different color and a white text label. The top segment is blue and labeled 'Strong Identity'. The bottom-left segment is purple and labeled 'Least Privilege'. The bottom-right segment is green and labeled 'Continuous Authorization'. In the top-left corner, there is a blue rectangular box with the text 'Zero Trust' in white.

Strong
Identity

Least
Privilege

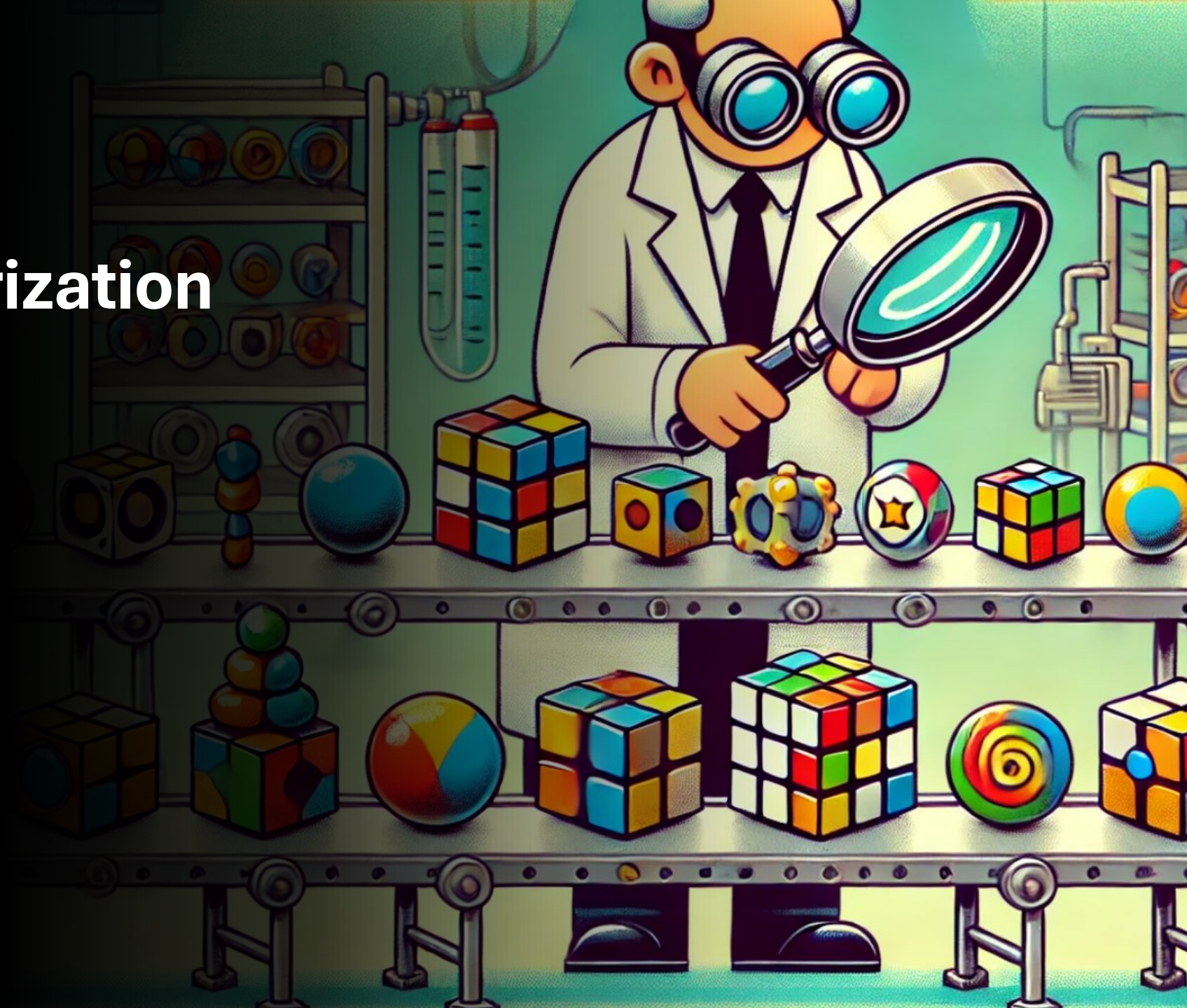
Continuous
Authorization



X.509
JWT

What is a “Strong Identity”

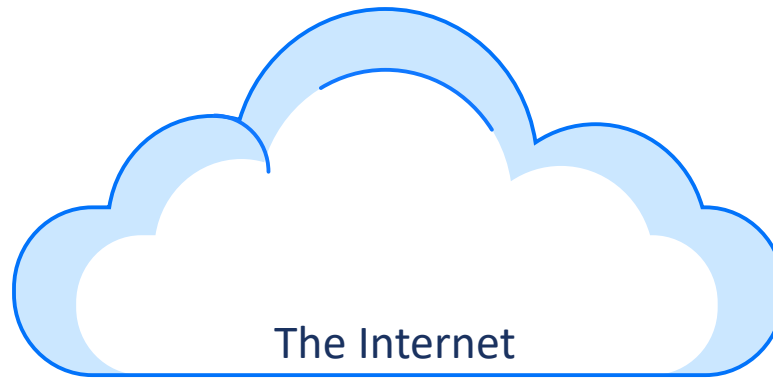
Continual Authorization



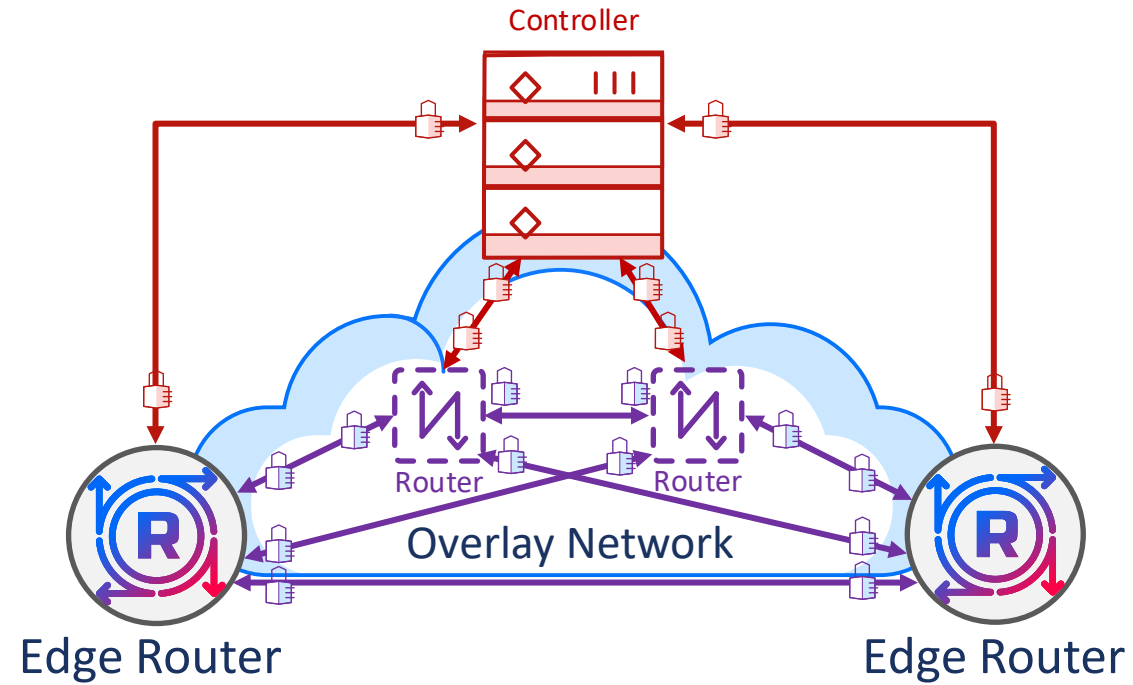


Least Privilege

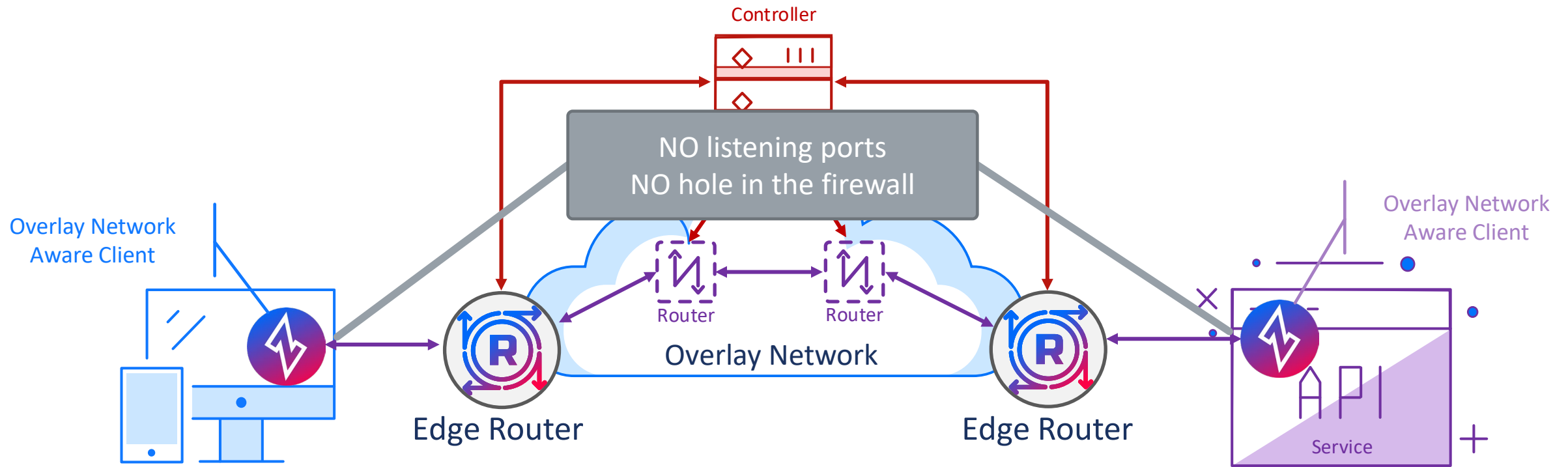
WHAT IS AN “OVERLAY NETWORK”?

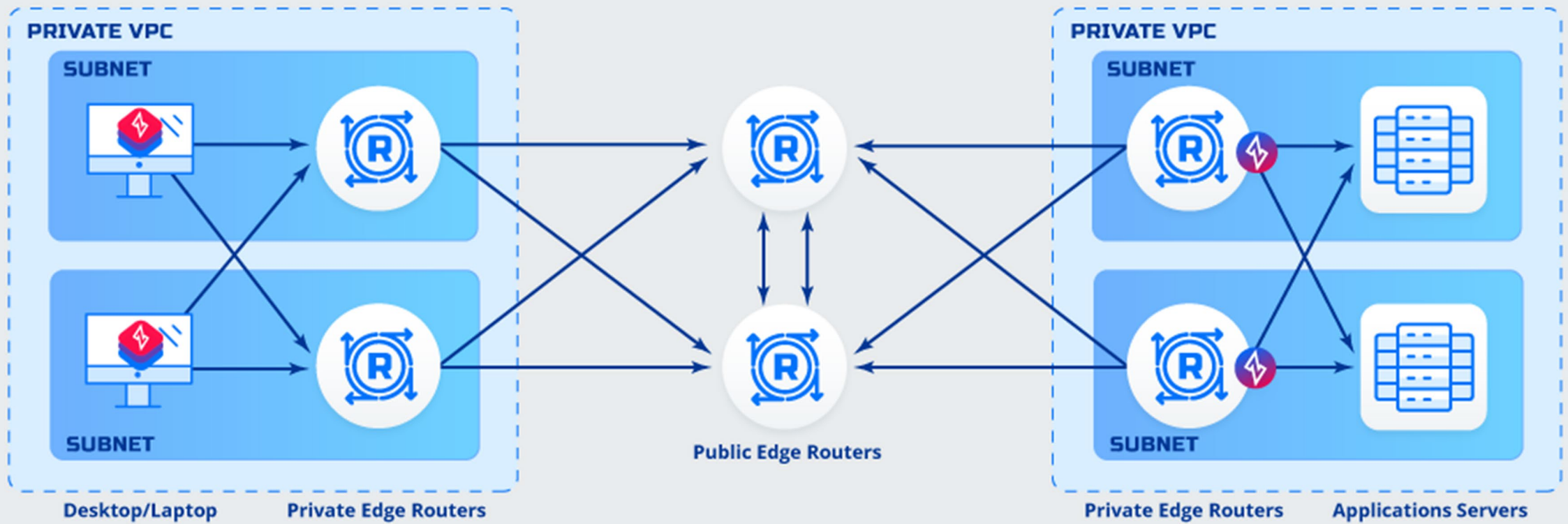


WHAT IS AN “OVERLAY NETWORK”?



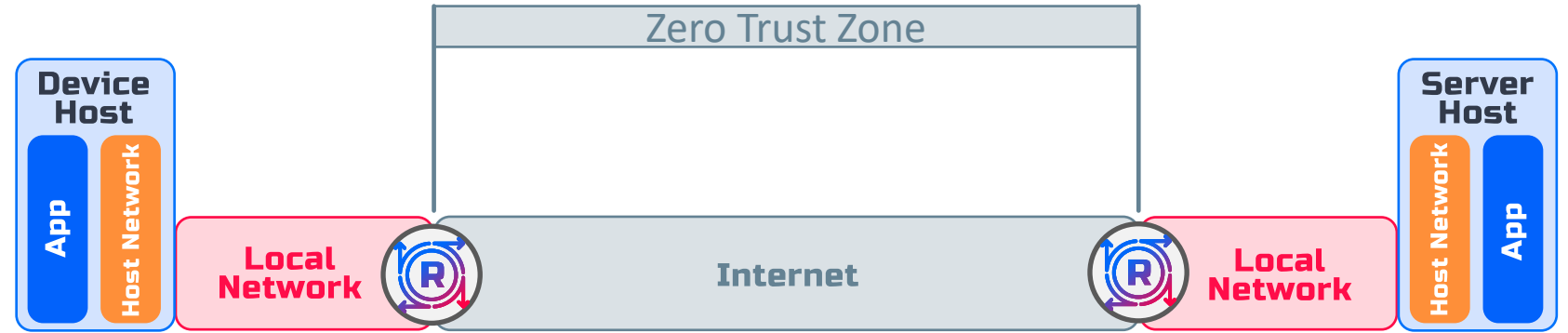
WHAT IS AN "OVERLAY NETWORK"?



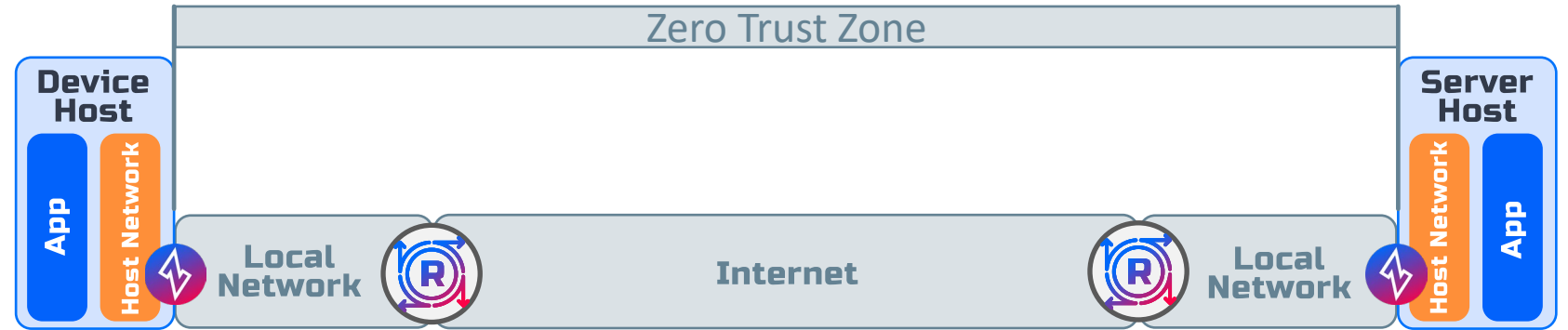


- Self-healing
- Optimal Routing
- Active Load Balancing

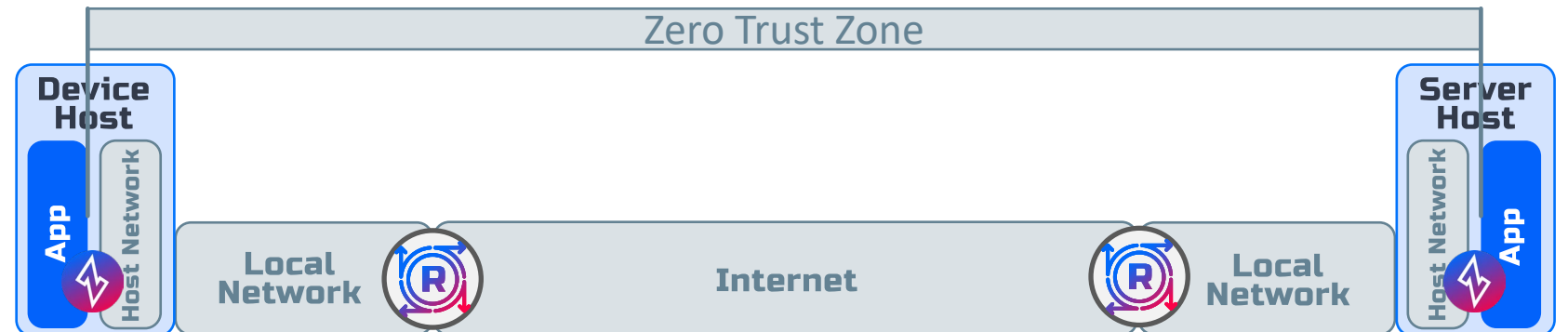
ZTNA: ZeroTrust Network Access



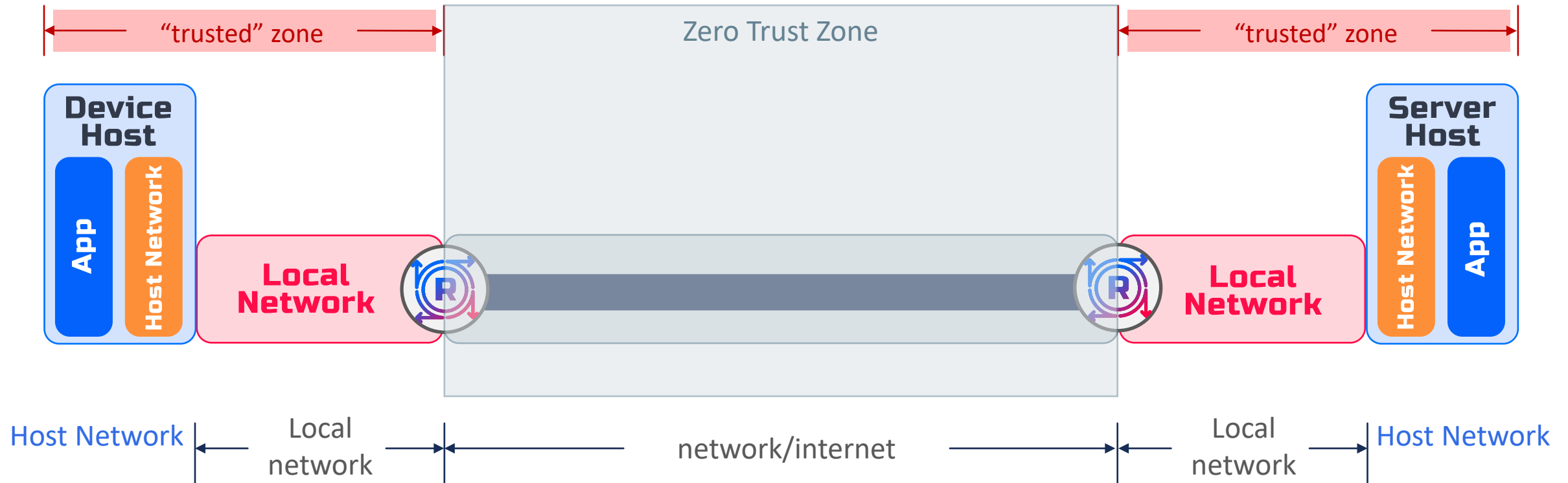
ZTHA: ZeroTrust Host Access



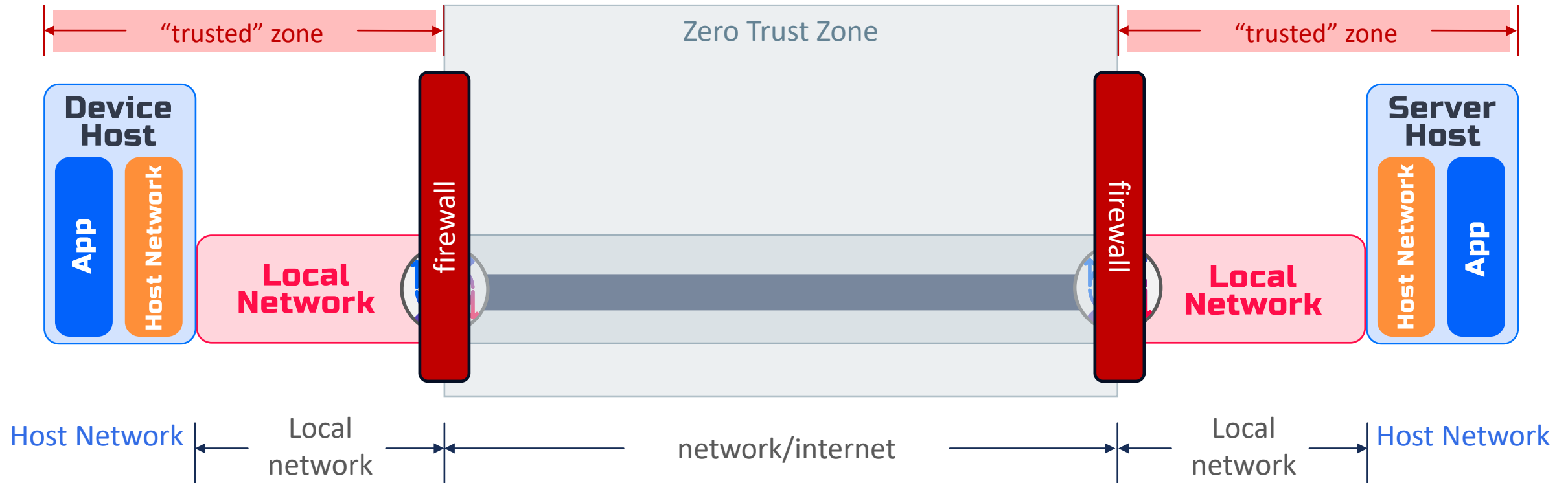
ZTAA: ZeroTrust Application Access



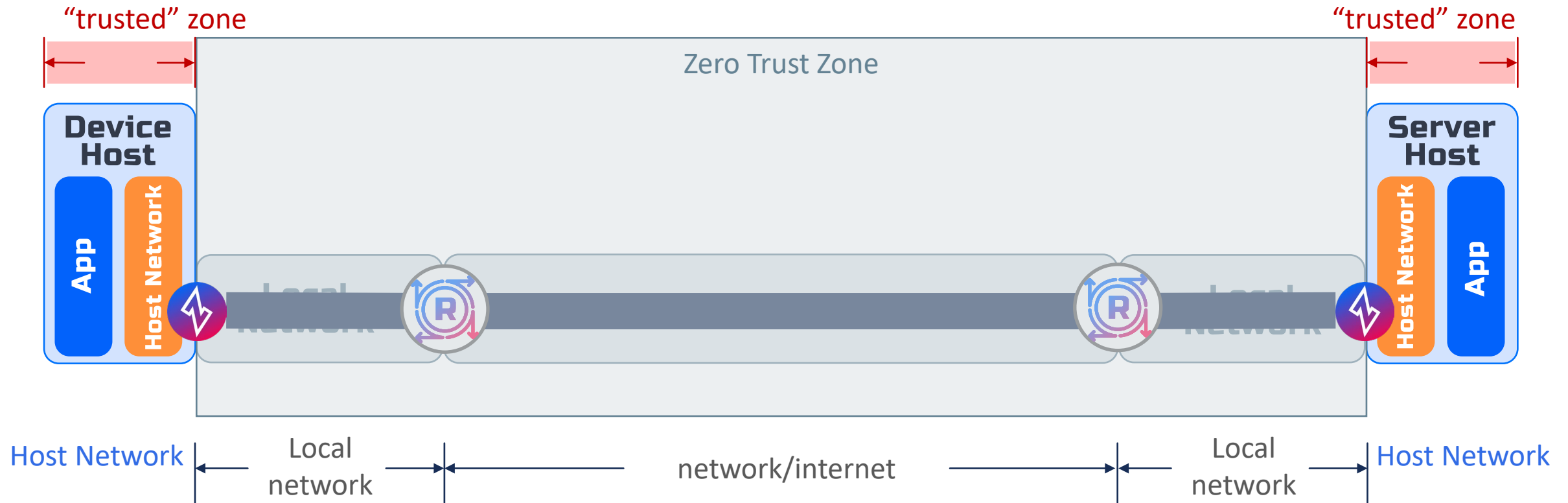
ZTNA: ZeroTrust Network Access



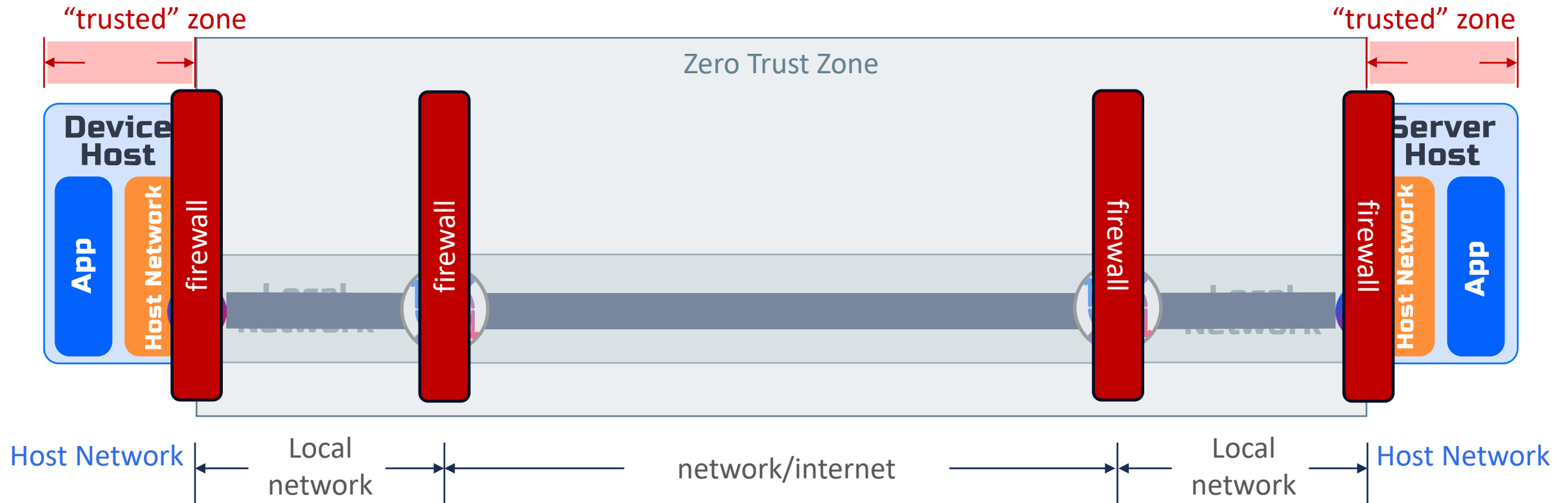
ZTNA: ZeroTrust Network Access



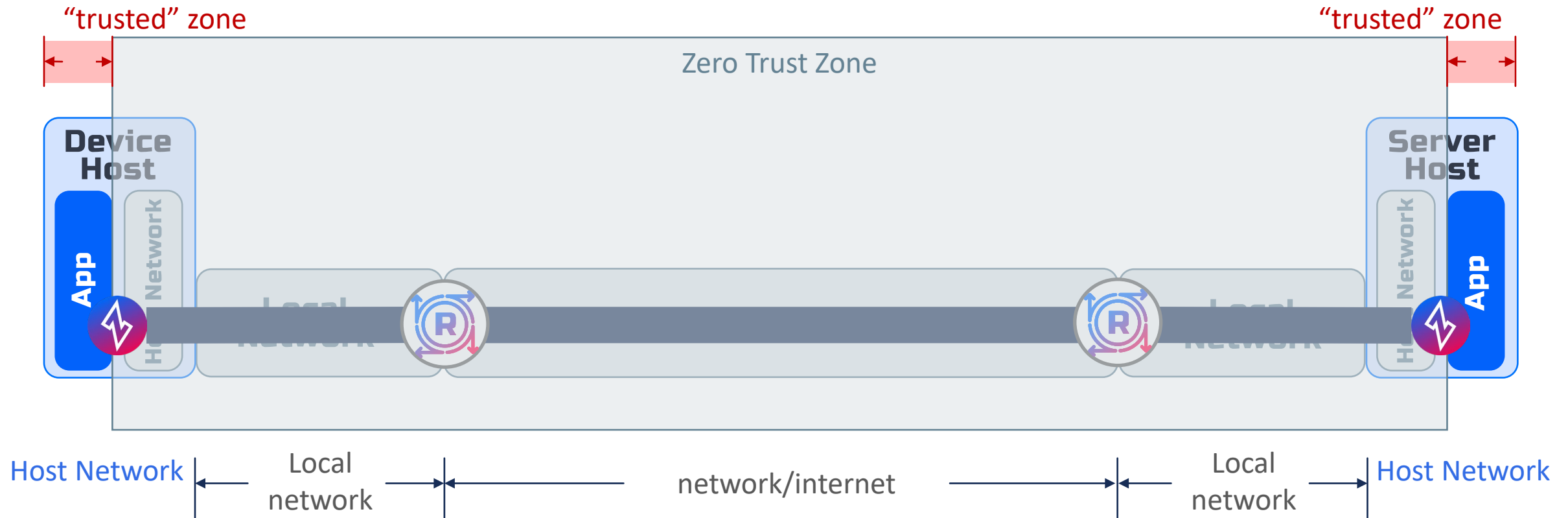
ZTHA: ZeroTrust Host Access



ZTHA: ZeroTrust Host Access

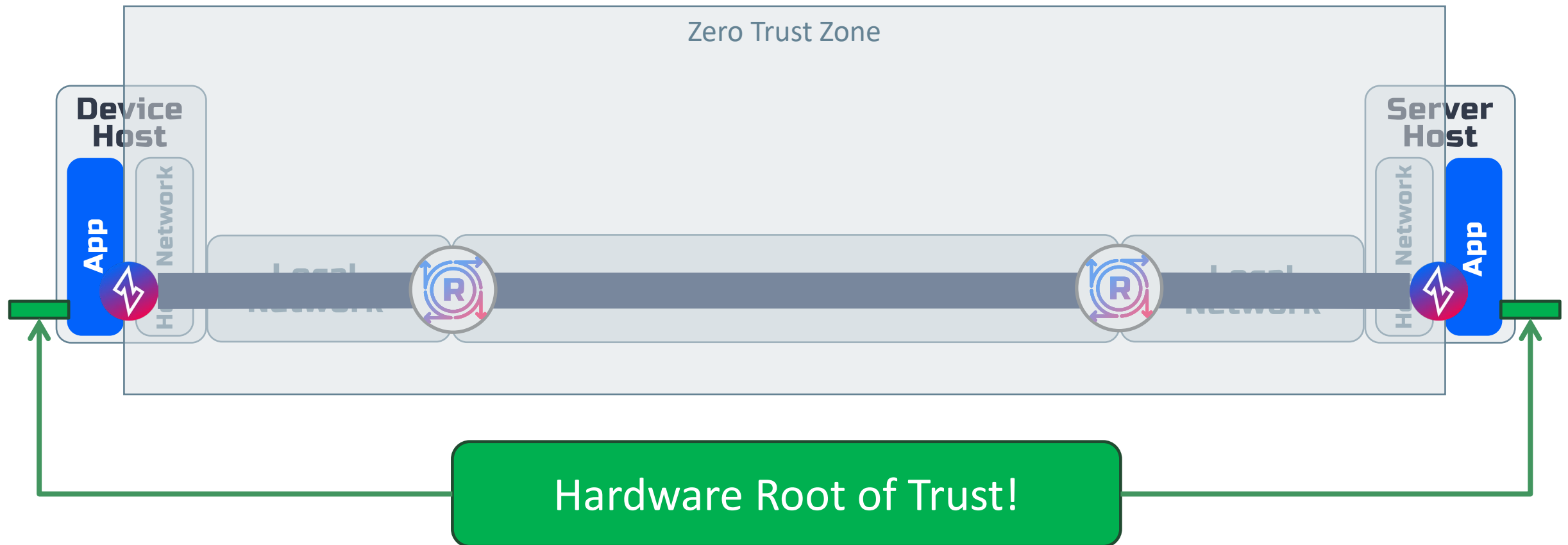


ZTAA: ZeroTrust Application Access



ZTAA: ZeroTrust Application Access

WITH HARDWARE ROOT OF TRUST



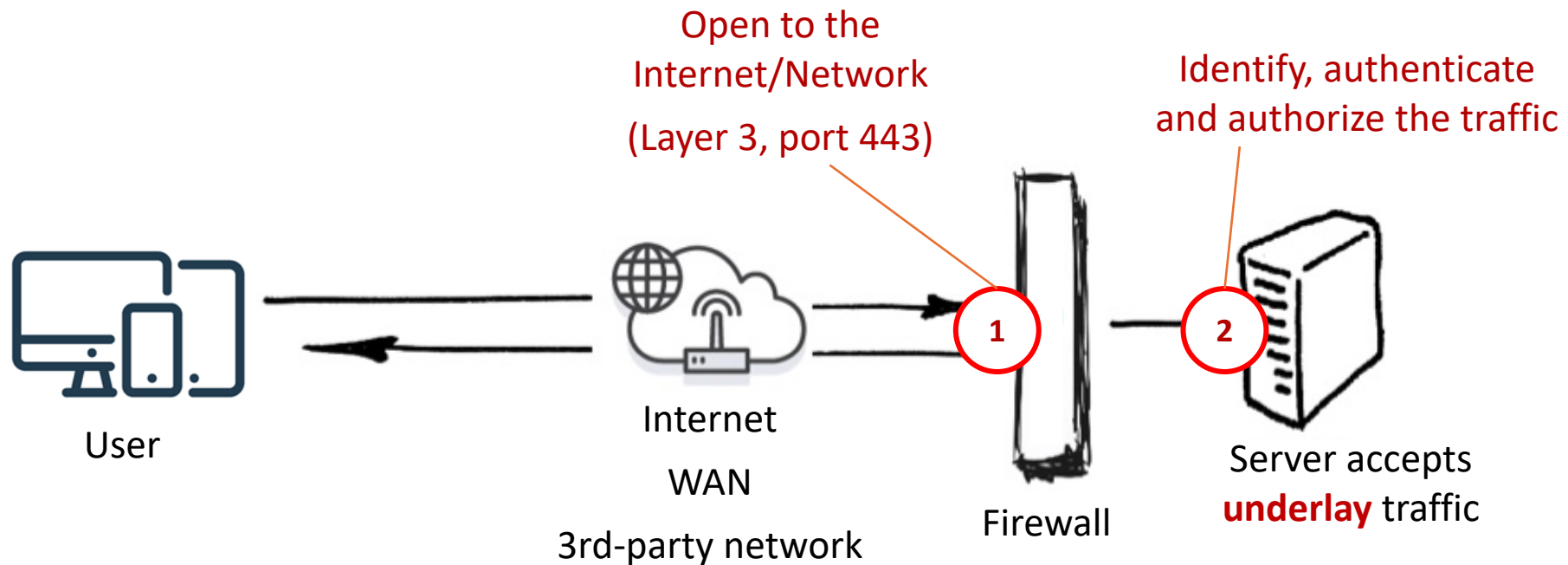


Authorize Before Connect (AB4C)





Even a single open inbound port is too many



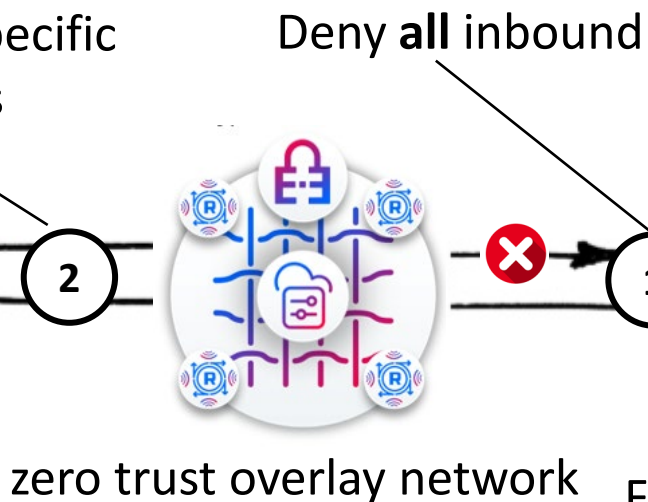


The Zero Trust Model that Works

Authenticate and authorize identities **before** to specific network services



User



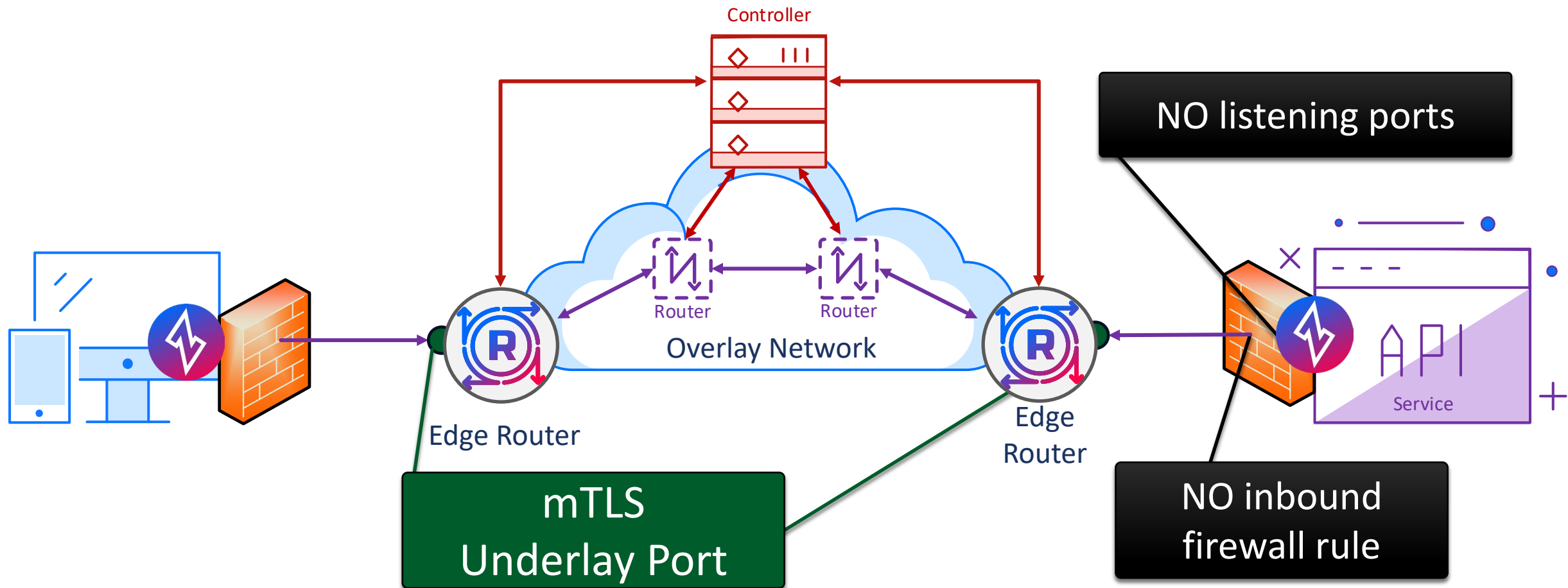
Traffic is authenticated and authorized **before** traffic is sent to the server



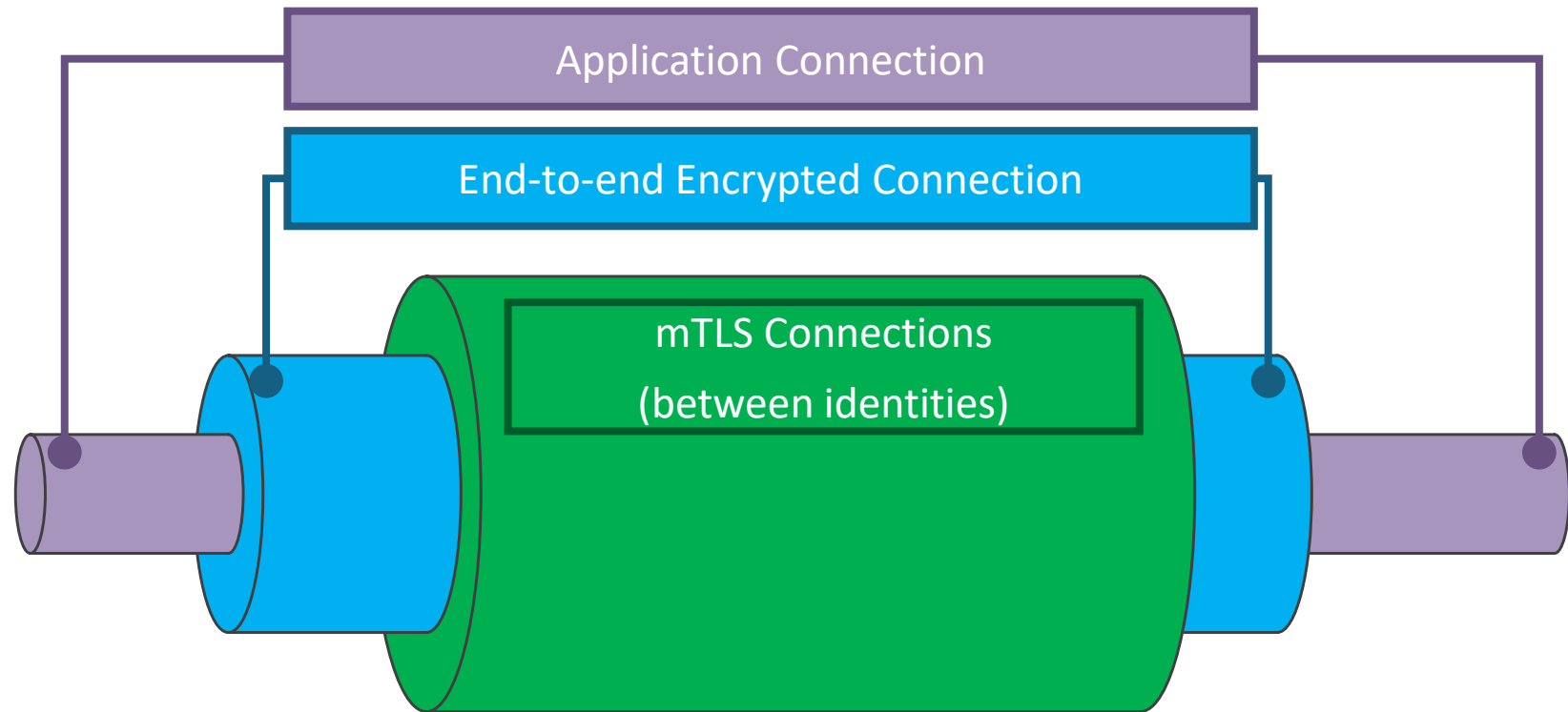
Server accepts **overlay** traffic

AB4C = *Authenticate Before Connecting*

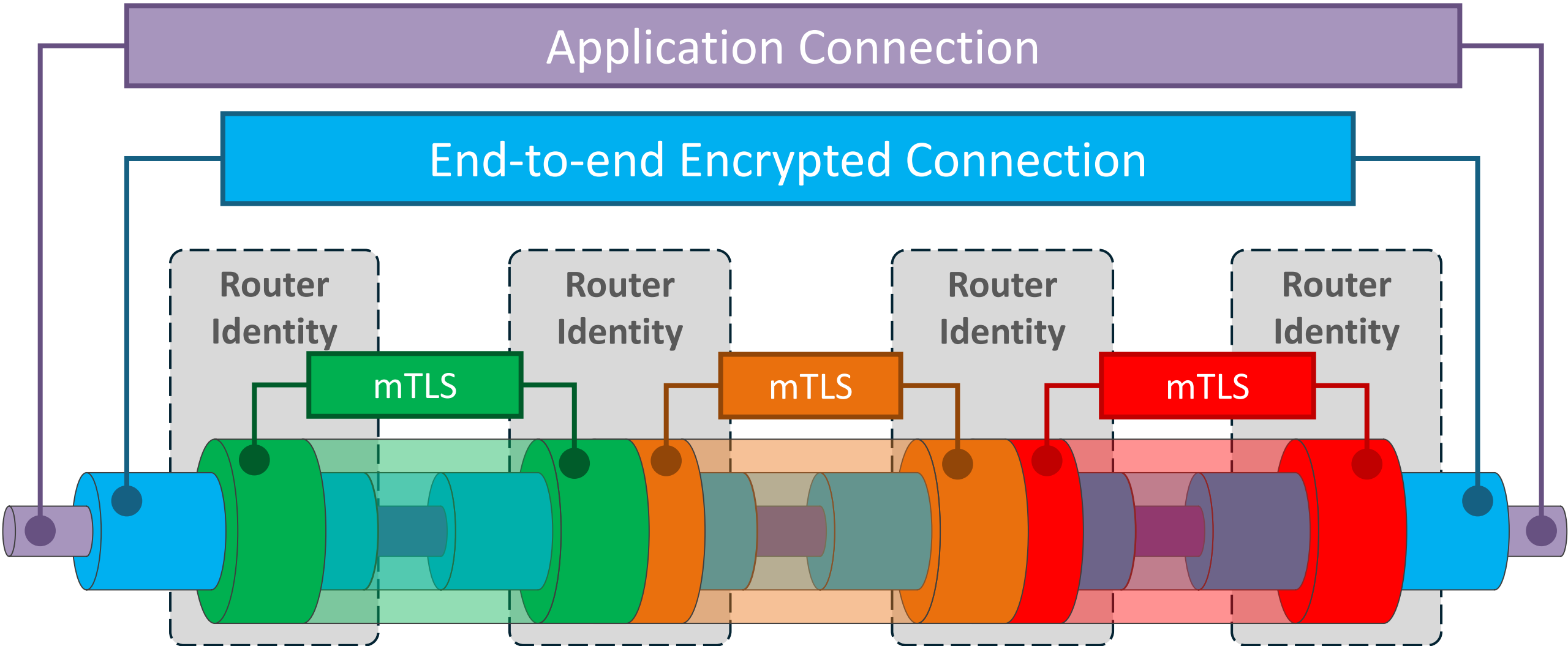
ZERO TRUST CONNECTION



End to End Encryption



Zero Trust End-to-End Encryption




```
operation == "MIRROR_Y":  
    mirror_mod.use_x = False  
    mirror_mod.use_y = True  
    mirror_mod.use_z = False  
operation == "MIRROR_Z":  
    mirror_mod.use_x = False  
    mirror_mod.use_y = False  
    mirror_mod.use_z = True
```

```
#selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob))  
mirror_ob.select = 0  
= bpy.context.selected_objects  
data.objects[one.name].select  
  
print("please select exactly")
```

-- OPERATOR CLASSES -----

LOOKING AT CODE!

ENOUGH WITH THE PICTURES

```
types.Operator):  
    X mirror to the selected  
    object.mirror_mirror_x"  
    X"
```

All Local

NO SECURITY



OPEN PORT: 18080
NO TLS!

nosecurity-
server



Network

nosecurity-
client



COMMON.GO

```
func CreateServer() *http.Server {  
    svr := &http.Server{}  
    mux := http.NewServeMux()  
    mux.Handle("/", http.HandlerFunc(index))  
    mux.Handle("/domath", http.HandlerFunc(mathHandler))  
    svr.Handler = mux  
    return svr  
}
```



COMMON.GO

```
func CreateUnderlayListener(port int) net.Listener {  
    ln, err := net.Listen("tcp", fmt.Sprintf(":%d", port))  
    if err != nil {  
        panic(err)  
    }  
    return ln  
}
```



NOSECURITY - SERVER.GO

```
func main() {  
    httpServer := common.CreateServer(nil)  
    ln := common.CreateUnderlayListener(common.InsecurePort)  
    log.Printf("Starting insecure server on %d\n", common.InsecurePort)  
    if err := httpServer.Serve(ln); err != nil {  
        log.Fatal(err)  
    }  
}  
  
const InsecurePort = 18080
```

↑
NO TLS



MTLS WITH SPIRE



Universal identity control
plane for distributed
systems

All Local

MUTUAL TLS WITH SPIRE



SPIRE
Server

SPIRE Agent

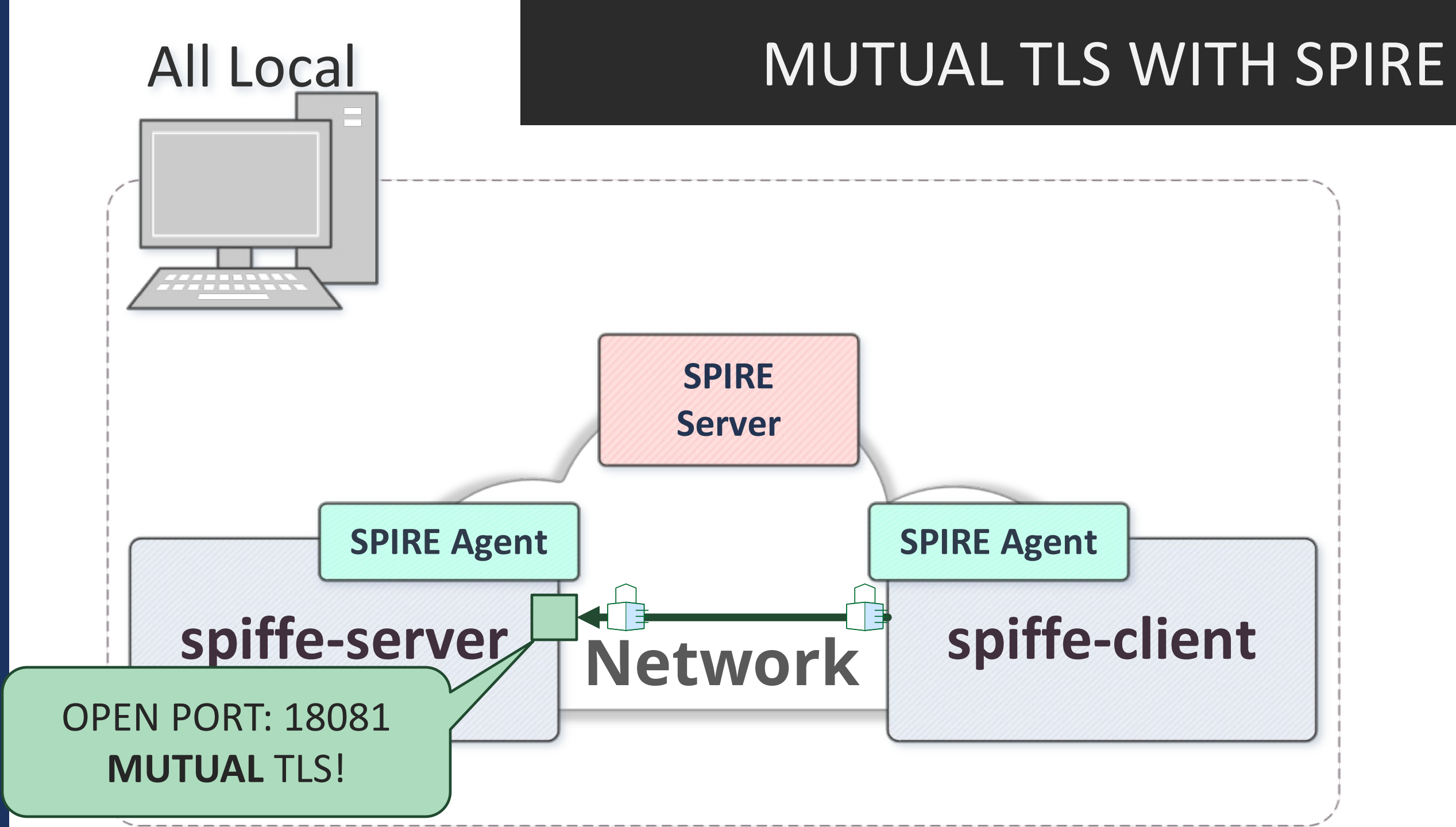
SPIRE Agent

spiffe-server

Network

spiffe-client

OPEN PORT: 18081
MUTUAL TLS!





SPIRE - SERVER.GO

```
func main() {  
    httpServer := common.CreateServer()  
  
    opts := workloadapi.WithClientOptions(workloadapi.WithAddr(common.SocketPath))  
    spire.ConfigureForMutualTLS(context.Background(), httpServer, opts)  
  
    ln := common.CreateUnderlayListener(common.SpireSecuredPort)  
    log.Printf("Starting server secured by SPIRE on %d\n", common.SpireSecuredPort)  
    if err := httpServer.ServeTLS(ln, "", ""); err != nil {  
        log.Fatal(err)  
    }  
}
```



SPIRE PACKAGE

```
func ConfigureForMutualTLS(ctx context.Context, server *http.Server, opts workloadapi.SourceOptions) error {
    source, err := workloadapi.NewX509Source(ctx, opts)
    if err != nil {
        panic(err)
    }

    // Create a `tls.Config` to allow mTLS connections, and verify/authorize
    // clients presenting certificates with SPIFFE ID `spiffe://example.org/client`
    clientID := spiffeid.RequireFromString(common.SpiffeClientId)
    tlsConfig := tlsconfig.MTLSServerConfig(source, source, tlsconfig.AuthorizeID(clientID))


    server.TLSConfig = tlsConfig
}
```



SPIRE - SERVER.GO

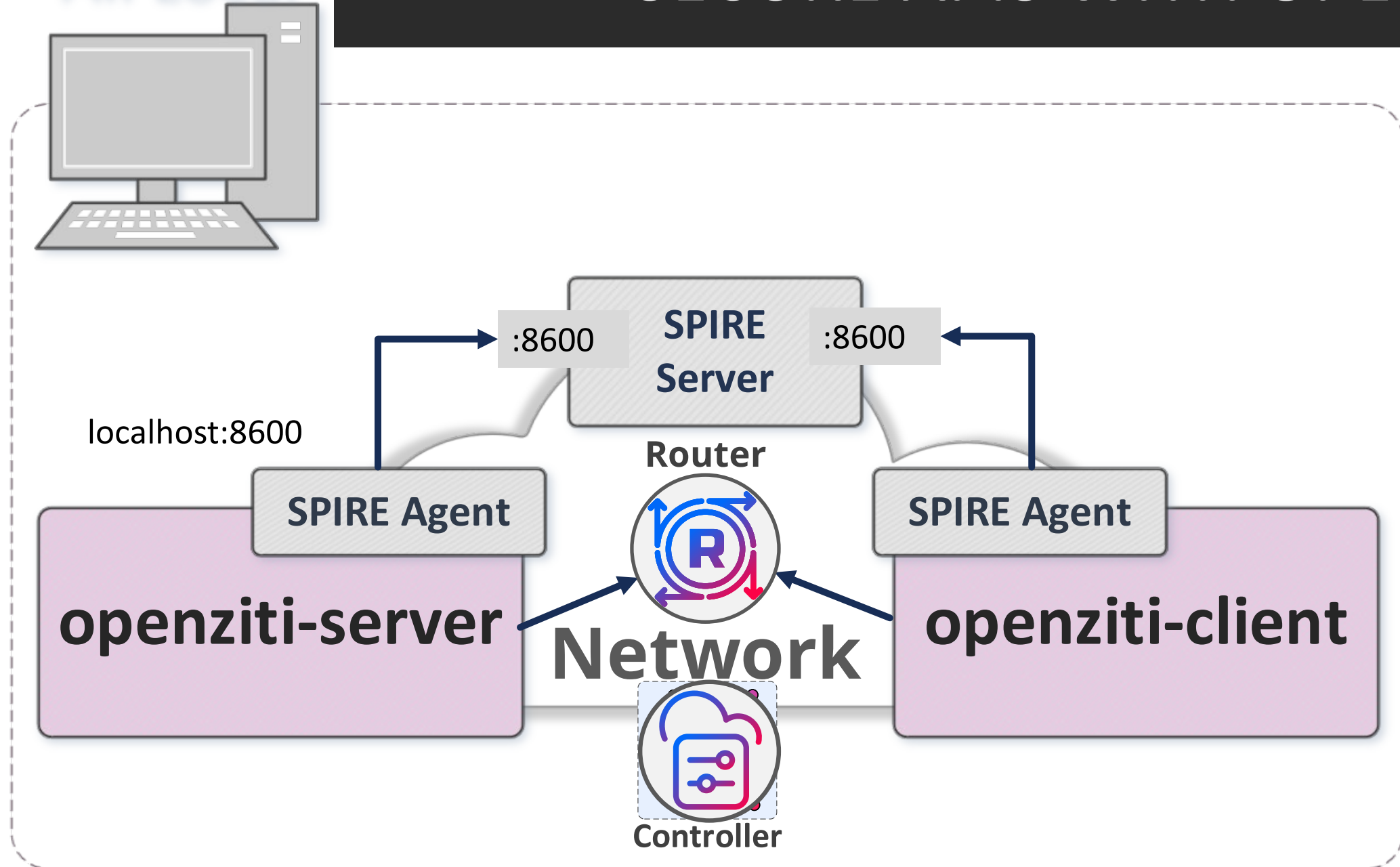
```
func main() {  
    httpServer := common.CreateServer()  
  
    opts := workloadapi.WithClientOptions(workloadapi.WithAddr(common.SocketPath))  
    spire.ConfigureForMutualTLS(context.Background(), httpServer, opts)  
  
    ln := common.CreateUnderlayListener(common.SpireSecuredPort)  
    log.Printf("Starting server secured by SPIRE on %d\n", common.SpireSecuredPort)  
    if err := httpServer.ServeTLS(ln, "", ""); err != nil {  
        log.Fatal(err)  
    }  
}
```

ServeTLS!!!



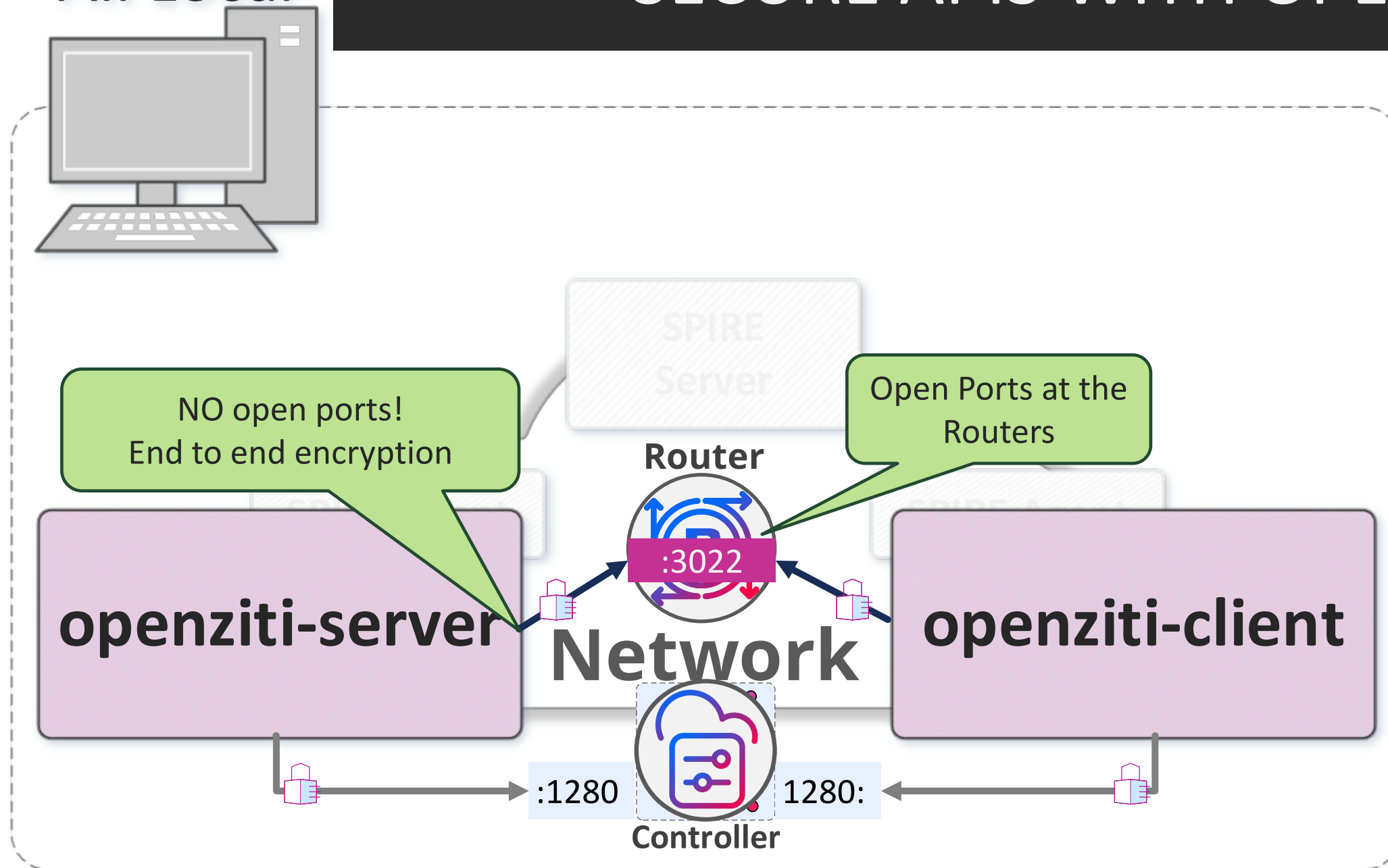
SECURE APIS WITH OPENZITI

All Local



All Local

SECURE APIS WITH OPENZITI





AUTHORIZING OPENZITI IDENTITIES

OPENZITI CONFIGURATION



```
signer=$(ziti edge create ext-jwt-signer zpire-ext-jwt zpire \  
--jwks-endpoint http://${SPIRE_OIDC}:8601/keys \  
--audience "spiffe://openziti/jwtServer")
```

```
authPolicy=$(ziti edge create auth-policy zpire-auth-policy \  
--primary-ext-jwt-allowed \  
--primary-ext-jwt-allowed-signers "${signer}")
```

OPENZITI CONFIGURATION



```
# create two identities for 'server' and 'clients'
ziti edge create identity service zpire-jwtClient \
--auth-policy $authPolicy \
--external-id "spiffe://openziti/jwtClient" \
--role-attributes demo-services-client
```

```
ziti edge create identity service zpire-jwtServer \
--auth-policy $authPolicy \
--external-id "spiffe://openziti/jwtServer" \
--role-attributes demo-services-server
```

OPENZITI CONFIGURATION



```
# create two demo services  
ziti edge create config openziti-and-spire-intercept.v1 intercept.v1 \  
  '{"protocols":["tcp"],"addresses":["openziti.spire.ziti"],  
  "portRanges":[{"low":443, "high":443}]}'  
  
ziti edge create service openziti-and-spire-service \  
  --configs openziti-and-spire-intercept.v1 -a demo-services
```

OPENZITI CONFIGURATION



```
# create two demo services
ziti edge create config openziti-and-spire-intercept.v1 intercept.v1 \
  '{"protocols":["tcp"],"addresses":["openziti.spire.ziti"],'
  "portRanges":[{"low":443, "high":443}]}'

ziti edge create service openziti-and-spire-service \
  --configs openziti-and-spire-intercept.v1 -a demo-services
```

OPENZITI CONFIGURATION



```
# authorize identities to bind
ziti edge create service-policy demo-services-bind-policy Bind \
  --service-roles '#demo-services' \
  --identity-roles '#demo-services-server'
```

```
# authorize identities to dial
ziti edge create service-policy demo-services-dial-policy Dial \
  --service-roles '#demo-services' \
  --identity-roles '#demo-services-client'
```



OPENZITI - SERVER.GO

```
func main() {  
    httpServer := common.CreateServer()  
  
    opts := workloadapi.WithClientOptions(workloadapi.WithAddr(common.SocketPath))  
    jwt, _ := spire.FetchJwt(common.SpiffeServerId, opts)  
    spire.ConfigureForMutualTLS(context.Background(), httpServer, opts)  
  
    ln := openziti.CreateOpenZitiListener(jwt, "openziti-only-service")  
    log.Printf("Starting server secured by OpenZiti on the OpenZiti overlay, no open port\n")  
    if err := httpServer.ServeTLS(ln, "", ""); err != nil {  
        log.Fatal(err)  
    }  
}
```



OPENZITI - SERVER.GO

```
func main() {  
    httpServer := common.CreateServer()  
  
    opts := workloadapi.WithClientOptions(workloadapi.WithAddr(common.SocketPath))  
    jwt, _ := spire.FetchJwt(common.SpiffeServerId, opts)  
    spire.ConfigureForMutualTLS(context.Background(), httpServer, opts)  
  
    ln := openziti.CreateOpenZitiListener(jwt, "openziti-only-service")  
    log.Printf("Starting server secured by OpenZiti on the OpenZiti overlay, no open port\n")  
    if err := httpServer.ServeTLS(ln, "", ""); err != nil {  
        log.Fatal(err)  
    }  
}
```



OPENZITI - SERVER.GO

```
func main() {  
    httpServer := common.CreateServer()  
  
    opts := workloadapi.WithClientOptions(workloadapi.WithAddr(common.SocketPath))  
    jwt, _ := spire.FetchJwt(common.SpiffeServerId, opts)  
    spire.ConfigureForMutualTLS(context.Background(), httpServer, opts)  
  
    ln := openziti.CreateOpenZitiListener(jwt, "openziti-only-service")  
    log.Printf("Starting server secured by OpenZiti on the OpenZiti overlay, no open port\n")  
    if err := httpServer.ServeTLS(ln, "", ""); err != nil {  
        log.Fatal(err)  
    }  
}
```


OPENZITI.IO

Proudly supported by



- Multi-cloud-native
- Secure-by-design
- Develop-once - deploy-anywhere
- Infrastructure-as-Code



CLINT DOVHOLUK

Developer / Zero trust evangelist

CLINT@OPENZITI.ORG



<https://openziti.github.io/>

Search for "OpenZiti"

twitter.com/openziti

github.com/openziti

openziti.discourse.group

github.com/openziti-test-kitchen

twitter.com/openziggy



HTTPS://
EVERYWHERE



Let's
Encrypt