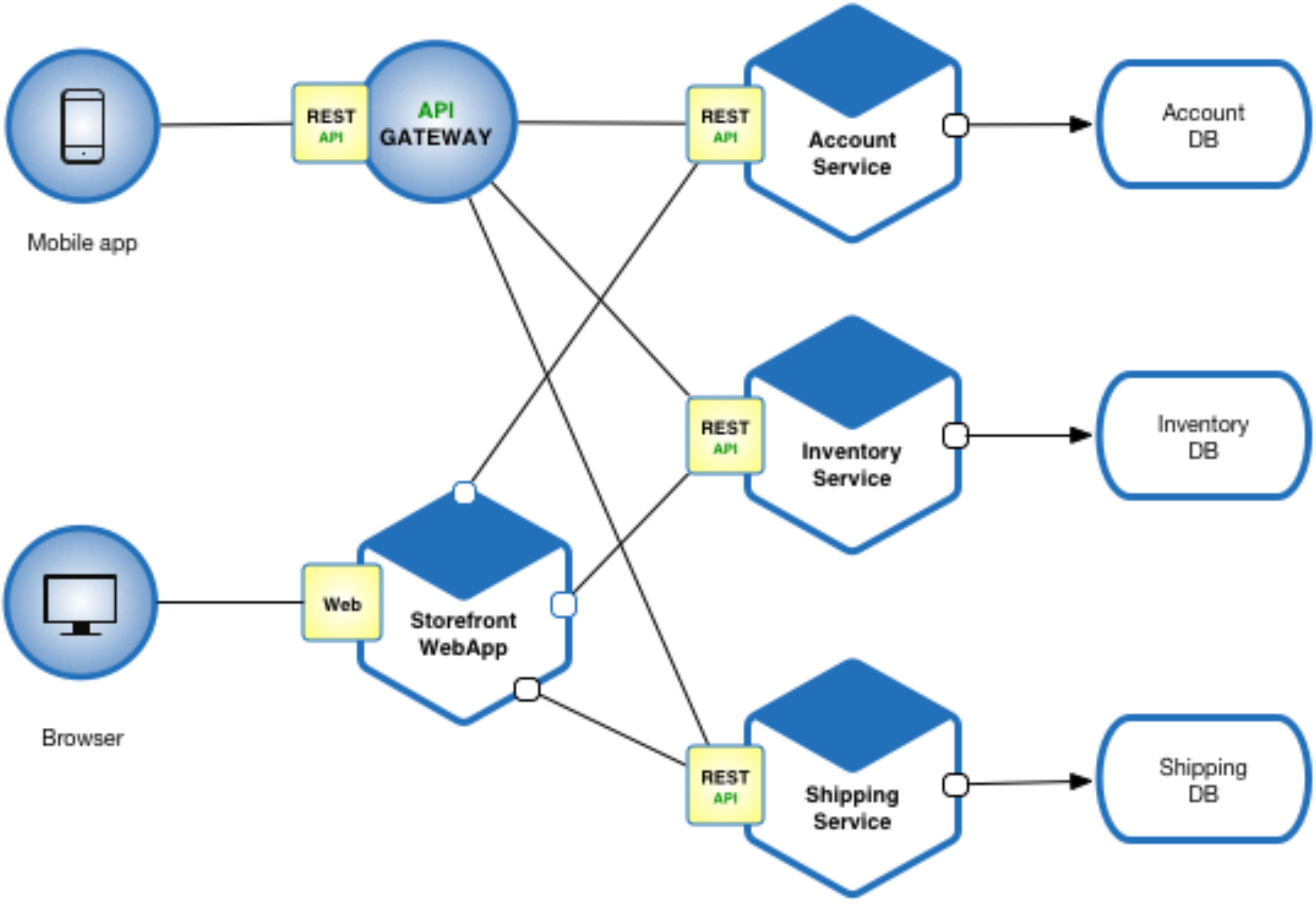
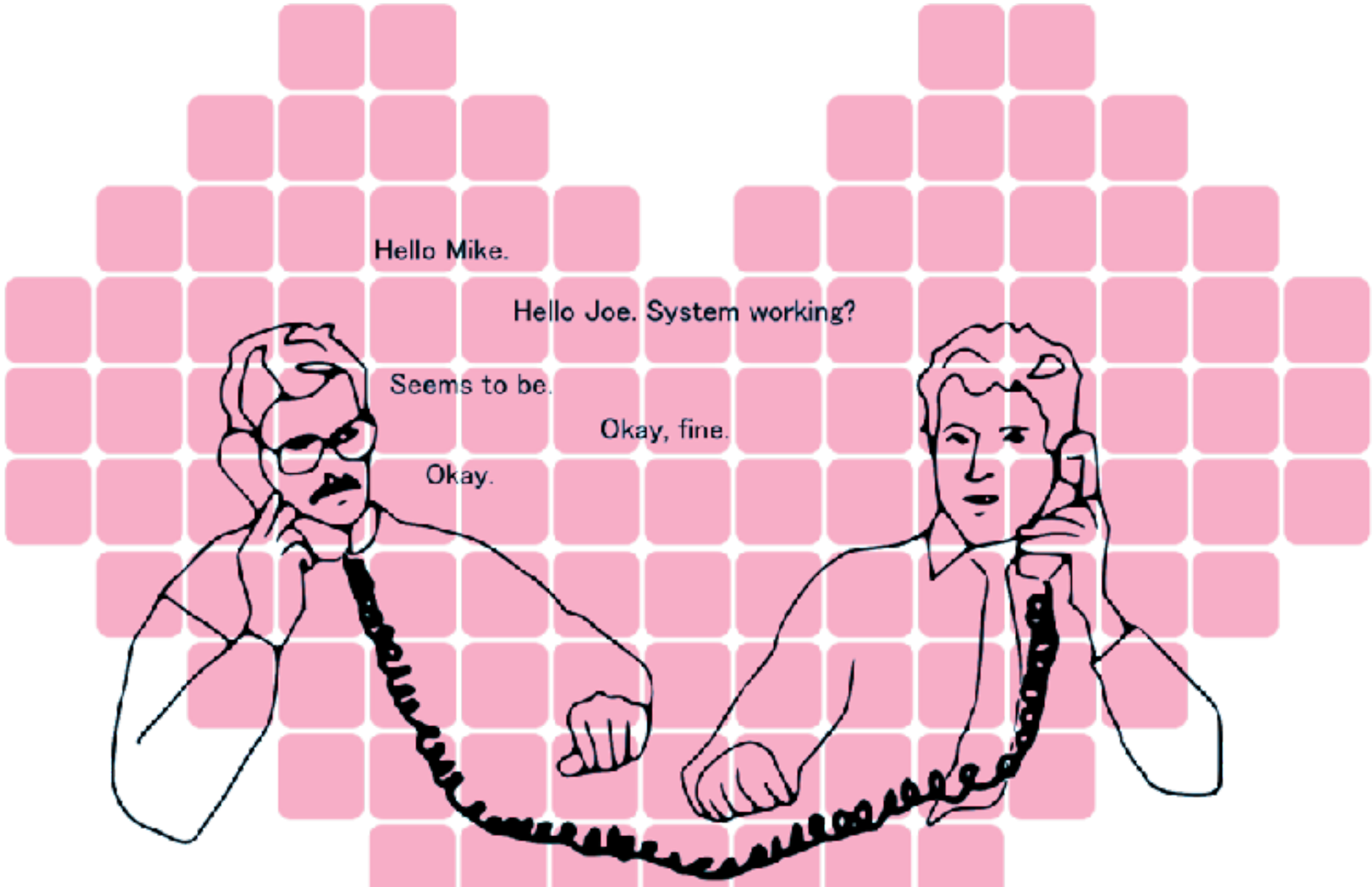
Security & Trust in a Services World

JEMURAI

Aaron Bedra
Chief Scientist, Jemurai
@abedra
keybase.io/abedra

In the beginning…


Mobile app
REST
API
API
GATEWAY
REST
API
Account
Service
Account
DB
REST
API
Inventory
Service
Inventory
DB
Browser
Web
Storefront
WebApp
REST
API
Shipping
Service
Shipping
DB


Hello Mike.
Hello Joe. System working?
Seems to be.
Okay, fine.
Okay.

Mike/Joe OTP

Does this change the way we approach security?

It certainly should!

In fact, it makes it “easier”

	Class	Package	Subclass (same pkg)	Subclass (diff pkg)	World
public	+	+	+	+	+
protected	+	+	+	+	
no modifier	+	+	+		
private	+				

+ : accessible
blank : not accessible

With a service architecture
we can draw our
relationships as they truly are

But we've got a lot to consider when it comes to security

Trust

noun
1.
reliance on the integrity, strength, ability, surety, etc., of a person or thing; confidence.
2.
confident expectation of something; hope.

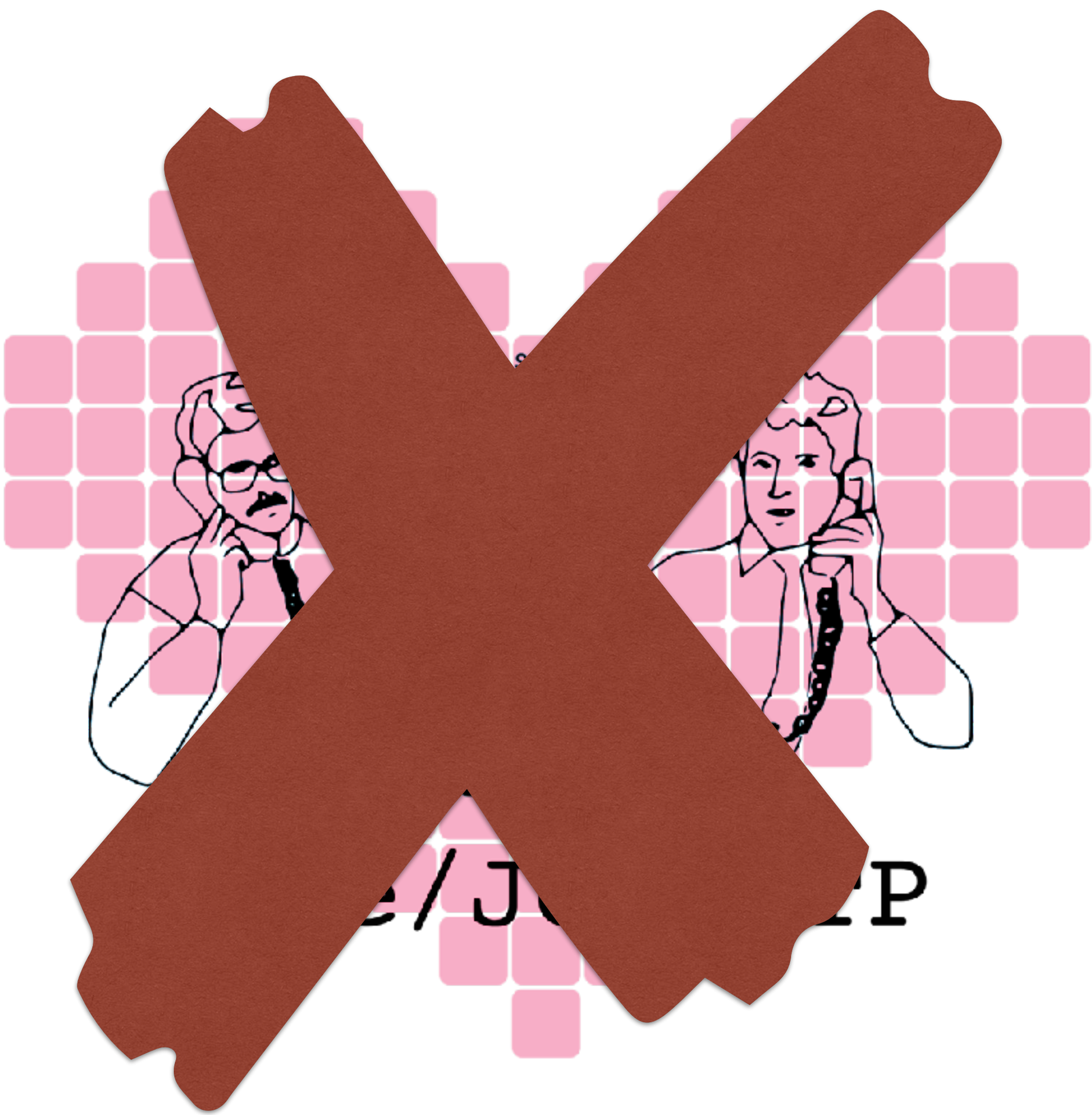
Trust != Authentication

Authentication speaks to identity

But does not address trust


SAY BLOCKCHAIN
AGAIN
imgflip.com

Some things to get out of the way



Trust is multivalent

In real life, once you learn someone's name, do you trust them with everything forever?

Of course not!

Our systems shouldn’t either

Trust is momentary and depends on context

And most importantly,
it can change

We will talk about classification later, but there are also levels of trust

Consider the following

Interesting questions

- Date of last penetration test?
- Vulnerable dependencies?
- Vulnerable container images?
- Known unmitigated findings?
- Deviations in behavior?

We should create layers of trust based on information available

This requires a more comprehensive security program

Yeah, but what do we do with it?

If someone you didn’t know asked you a deeply personal question, would you answer it?

What about someone you have known for years?

What if that person started asking really strange questions?

Would you alter your notion of trust?


NBC

Let’s pull it back to technology

We can shift to momentary trust

More questions?

- Who performed authentication?
- Do they agree you are who you say you are?
- What else do we know about you?
- Based on what we know, to what degree can we trust you?

```
{
  "last_penetration_date": "2017-04-26T16:24:44+00:00",
  "open_findings": true,
  "repository": "github.com/company/service",
  "dependency_file": "package.json",
  "vulnerable_dependencies": true,
  "current_container": "registry.local/service/latest",
  "container_vulnerabilities": true,
  "build_status": "failing",
  "classification": "private",
  "service_dependencies": ["sheep", "cheese"],
}
```

This information can and will change

Use it to determine if they meet your criteria for delivering information

In fact, publish your requirements as part of your service definition

Publishing trust requirements helps prevent unintended interruptions

Yeah, yeah, that’s nice, but you’re insane. We can’t do this!

Good point

I’m not here to convince you to improve security

I'll read about you in the news someday

Please stop thinking about this as a security exercise

It’s a design exercise

Because it’s what you do once you have this that truly matters

Service Classification

What types of data pass through a service?

Types of Data

- Public
- PCI
- HIPAA
- PII
- Internal
- Confidential

A service should be classified by the most sensitive data that passes through it

A service doesn’t need to store data to be classified

It just has to have access to it

How do we record classifications?

Use a service registry!

```yaml
apiVersion: v1
kind: Service
metadata:
  name: user-service
  labels:
    classification: private
spec:
  type: LoadBalancer
  ports:
  - port: 8888
  selector:
    app: user
```

This is a simple example, but you can plug this idea into any registry

What do we do with it?

Restrict the flow of data based on classification

Scenario

The cardholder data service is classified as PCI

The profile service is classified as PII

Should the cardholder data service return PCI scoped data?

NO!

It should only pass what it is allowed to based on the caller's classification

Using only a single interface

This means filtering responses based on classification

DEMO

```
func buildResponse(classification string, user User) User {
   switch classification {
   case "public":
      return User{
         Username: user.Username,
         First:    user.First,
         Last:     user.Last,
         Email:    user.Email}
   case "private":
      return User{
         ID:       user.Id,
         Username: user.Username,
         First:    user.First,
         Last:     user.Last,
         Email:    user.Email,
         Password: user.Password}

   }
}
```

How do we know the classification of the caller?

```
func getServiceClassification(service string) string {
   fmt.Println("Getting classification for", service)
   config, err := rest.InClusterConfig()
   if err != nil {
      log.Fatal(err)
      return "public"
   }

   clientset, err := kubernetes.NewForConfig(config)
   if err != nil {
      log.Fatal(err)
      return "public"
   }

   s, err := clientset.Core().Services(“default")
                .Get(service, metav1.GetOptions{})
   if err != nil {
      log.Fatal(err)
      return "public"
   }

   return s.GetLabels()["classification"]
}
```

Yeah, but how do we know the classification of the caller?

This is where trust comes into play

Without some level of authentication this is very difficult

Or potentially impossible

You could use JWT

```
{
  "typ": "JWT",
  "alg": "HS256"
}
{
  "iss": "token-service",
  "service": "frontend",
  "jti": "1e7e906b-9c78-47dd-bc50-4b1d77ccab55",
  "iat": 1524758983,
  "exp": 1524762583
}
```

```
{
  "typ": "JWT",
  "alg": "HS256"
}
{
  "iss": "token-service",
  "service": "frontend",
  "jti": "1e7e906b-9c78-47dd-bc50-4b1d77ccab55",
  "iat": 1524758983,
  "exp": 1524762583
}
```

Or pass the token of the caller to a lookup service

```
func getApplication(conf *conf, token *string) (string, error) {
   var application string
   query := "SELECT application from tokens where api_token=?"
   stmt, err := conf.Connection.Prepare(query)
   err = stmt.QueryRow(token).Scan(&application)
   if err != nil {
      return nil, err
   }

   return application, nil
}
```

Once you have identified the classification you can produce the appropriate response

Make sure you log everything about how you produced the information

What does this provide

- An audit trail of calls with the classification of the caller
- An audit trail of the classification of data that was returned by the callee to the caller
- A guarantee that data of specific classifications only reached designated locations

Why is this important?

If you don’t understand the flow of data, how can you protect against attack?

If you don’t understand the flow of data, how can you determine the depth of a breach?

If you log accurately, you can produce precise data flow models

```
{
  "timestamp": “2018-04-26T16:24:44+00:00”,
  "caller": "frontend-service",
  "callee": "user-service",
  "caller-classification": "public",
  "response-classification": "public",
  "source": "jwt"
}
```

Which lets you build accurate threat models

But also provides evidence for auditors

As you can see, we've got some work to do


DON'T
PANIC

A lot of these ideas have yet to materialize

But if we want to start taking security seriously, this type of discipline is important

If we do this right more than security falls out

Doing this right benefits architecture, operations, and business intelligence

Parting thoughts

Questions?