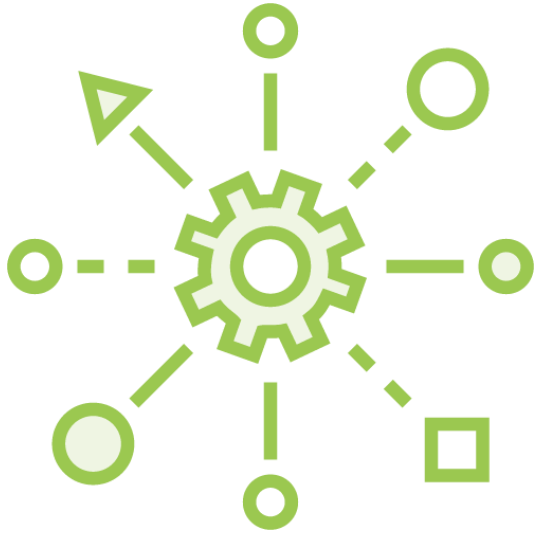
Security in ETL Deployments

Chris B. Behrens
SOFTWARE ARCHITECT
@chrisbbehrens

A Different Kind of Section

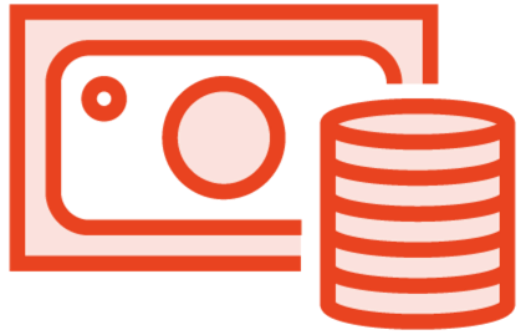
Security Considerations for Transferring Files

Getting data from big companies...



And doing stuff with it their IT couldn't do

Dr. No



Only big companies can afford a Dr. No

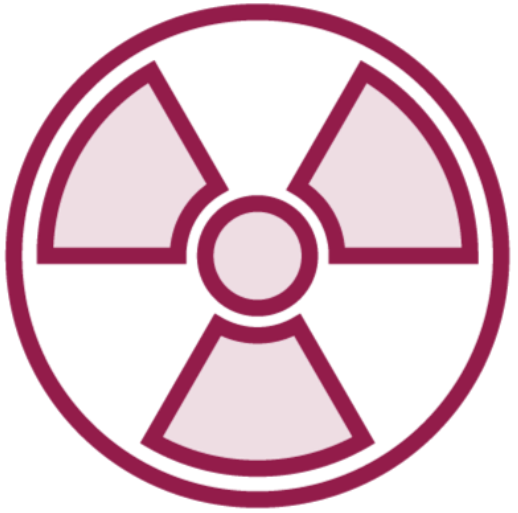
Dr. Maybe

Security against What?

The foremost security question

A huge company

Whose files detailed their user population



The Social Security Number

Why does a wellness program need a user's social security number?

“If I have your SSN, I can call the IRS and confess to tax fraud and hang up. And you’ll spend the next three years in audits.”

Chris B. Behrens, trying to scare a customer out of using SSNs

How Do We Deal with It?

"We don't accept Social Security Numbers as a matter of policy."

"We have a standard data format we export, and we can't change it."

"I can create a custom export package for you."

This rarely worked.

Strategy one: Don’t receive it in the first place.

When You Have to Receive It

The data is coming tonight, whether you want it or not

So we have to figure out how to handle it securely

Our first vector – interception in transit

Transport Security Via Secure Email

They're emailing you the files

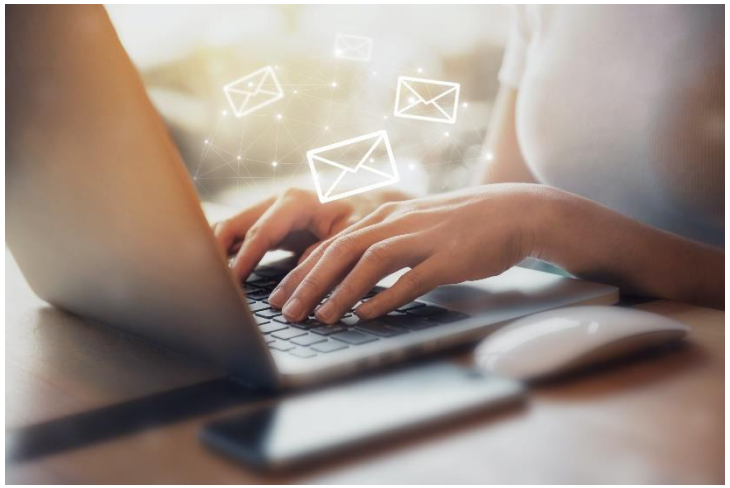
Sometimes "email" isn't actually email

Sometimes it's actually HTTPS – which is a good thing

Because the security of the sensitive data is protected by the security of the connection

Strategy two: Send files using transport-level security.

Vanilla Email Transport



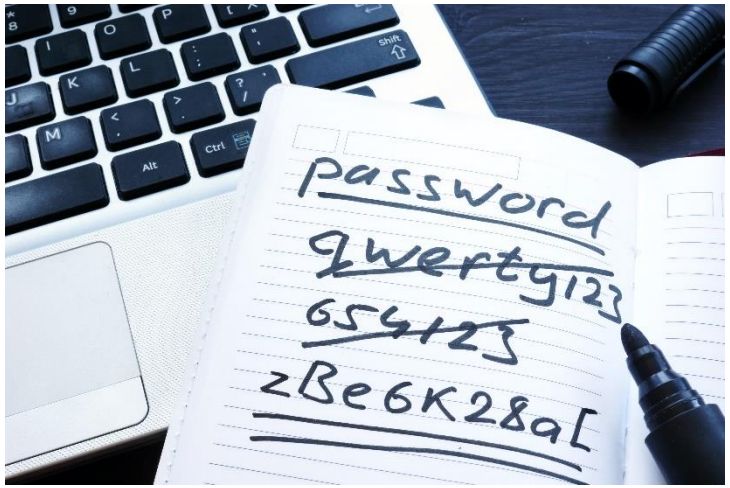
Sometimes, people do just totally send you unencrypted data via plain vanilla email

Regular email is not secure without some work

On both ends

Data sent this way should be assumed to have been compromised in transit

Excel Password (Non)Protection



"Don't worry, I password protected the Excel file."

This is merely politeness-level security

Maximum length = 255 characters

Excel passwords can be easily brute-forced

Just like our package passwords

Transport and Message-level Security

So...what's the right way to do it?

Transport security secures the communication channel

Like a secure train car

Message level security secures the message

So that it can travel over insecure channels

How to Do It Right

1. We generate a public and private key pair. We share the public key with the people sending us the data, and keep the private key
2. Each night as the final step before transfer, the sender used GPG and our public key to encrypt the data
3. The sender transfers the file
4. Using our private key, we decipher the sent file and then perform our ETL on it

Strong Encryption

This type of encryption is extremely strong

So strong, in fact, it was considered dangerous to share

And classified as a *munition* for the purposes of export

Strategy three: Perform your files transfers with message-level security.

Implementations of Each Approach

Secure Method	Insecure Equivalent	Security Type
Encrypted email message	Plaintext email	Message
SMTPS	SMTP (Plain vanilla email)	Transport
Encrypted file over FTP	Plaintext file over FTP	Message
FTPS, SFTPS, FTP/ES	FTP	Transport

File Transfer Security Wrap-up

Transport-level or

Message-level

Make this decision early and pay close attention to your process

Security Considerations for Data at Rest

Encryption at Rest

Transport-level Security

Archive files are stored in plaintext

SSN is available to be compromised

Message-level Security

Archive files are stored in ciphertext

SSN is unobtainable without the private key

Encrypting Data at Rest

Encrypt your data at rest, or leave it encrypted

Encrypt your sensitive columns

And definitely ensure that your incoming and archive files are encrypted at rest

Using Our Message Level Approach

1. A trigger encrypts the file
2. You decrypt the file as the first step in your ETL
3. When complete – you simply transfer the original encrypted file to the archive folder

Transparent Data Encryption (TDE)

Encrypt at a lower level

At the I/O level

Management of keys is transparent

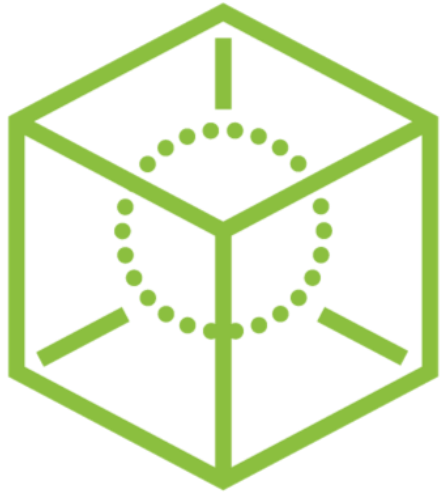
Transparent, but not too transparent

Not so transparent that it makes it easy for an attacker

Where Are My Keys?

Keep your keys secure and safe

Never store your key on the same server you will be performing operations on



Again, transparent, but not TOO transparent

Non-sensitive data evolves to become sensitive over time.

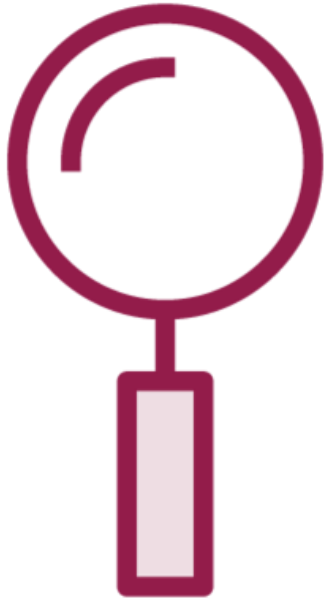
Vulnerability Sneaks In

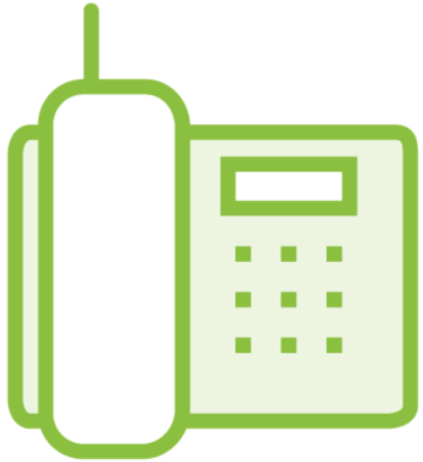
Our data was initially non-sensitive

But when combined with other data

It BECAME sensitive

A Nightmare Scenario





“This is John from Dr. Johnson’s office - we’re updating our records, can you quickly confirm your address for me? And the last four digits of your social security number?”

There’s no such thing as insensitive data.

A Problematic Deployment Pattern

Triggering the Build

1. Check-in to version control
2. Trigger build
3. Build checks out the content from version control
4. Build acts on that content

Triggering the Build

1. Data provider sends data
2. Data file gets checked into version control automatically
3. Check-in triggers build
4. Build checks out latest data file
5. Build executes package against data file

I've Made a Huge Mistake

And I realized it pretty quickly

I wiped out the repo

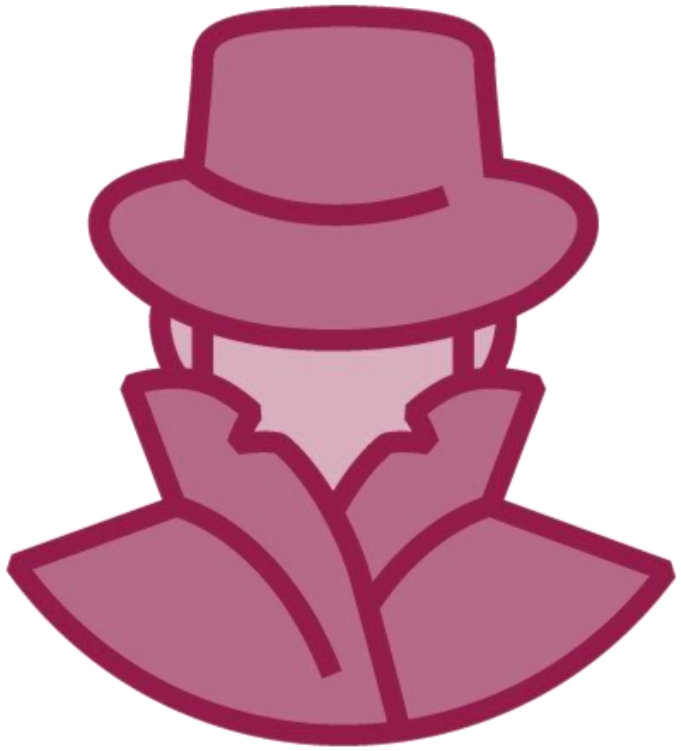
And physically shredded the drive

Don't put sensitive stuff in version control

When Your Stuff Gets Compromised

https://bit.ly/2LIsZGR

A Big Bill



EC2 = a service “that allows users to rent virtual computers on which to run their own computer applications.”

Your bill = $14k

The Github-stored creds were compromised almost instantly by bots searching public repos for keywords like ‘password’

And used to serve up malware

Don’t Store It at All

Two things:

- 1. You may need to revisit a file for troubleshooting or other purposes
- 2. Storing the data introduces the risk of an attack

Maybe it’s worth storing

Maybe not

Make a thoughtful decision

Hash It

An irreversible crypto operation

Like how passwords are stored

Hashing SSNS

If we hash the SSN we receive...

If somebody gets a hold of it, it's worthless*

Especially if we SALT the HASH – add a random string to the end to make it harder to compute

Who Doesn't Like Salted Hash?

In years of pitching salted hash as a security solution...

Hardly anyone was technically capable of implementing it

The Domain Is Too Small

Domain = “what the possible values are”

Rainbow tables – a list of hashes computed ahead of time for all possible values in a given domain

The rainbow tables for all possible SSNs is pretty small

But the salting makes it much larger

“One of these days, Congress is going to pass a law making this illegal. And all of these systems that rely on SSN as a key are going to fall to pieces.”

Chris B. Behrens, still trying to scare a customer out of using SSNs

What I Would Say Today

I got lucky

Privacy and security are a bigger deal now than they were then

Dr. No More SSNs

The Yuppie Nuremberg Defense

"I have to pay my mortgage"

Stick to your guns – your career will be longer than you think

Summary

Security

What it takes to be able to say:

- "This is a secure"
- In transit
- At rest

Why data doesn't belong in version control

What to do when you have to accept sensitive data