Who are you?

1. **Who are you and why are you able to talk about this?**
   I am the IT Director of a big company in the public sector. I am very involved in the application development we do.

What happened?

1. **What happened? What is your description of what happened?**
   The most important premise is that we are using large commercial cloud services. PAS services or platform as a service. When you are using these kinds of services you can create resources in these PAS services without using your own domain and certificates. Then you get assigned a subdomain name to the resource and you keep the subdomain before the first dot until the resource is taken down. And you have no control over IP restriction, it's all maintained by the PAS service and its all standard out of the box setup. And that is what everybody does. **“And it's not unusual for your team to do this, you create resources all the time?”**
   Yes. Totally not out of the norm. But you have to remember that everything that is hosted by a cloud vendor is on the internet. You can use authentication and normal security measures on it. But then you are going a little further. And this is what people do every day. They spin up resources that have externally exposed resources that are exposed to the internet. Then it is just a question about what security measures you do when you are developing something.
   And the mistake people make is that this is just “dev” or this is just “test” we are not in “prod”. We haven't told anyone about this. No one knows that this resource is out there. But you don’t have to tell anyone it’s out there because the chinese have already scanned it and put it on a list. “**Yes. There are so many botnets out there just looking for these types of sites or resources”** Yes! And as soon as you have created this resource with the public cloud vendor then that endpoint is registered by every hacker group in the world. Everyone just scanned it and we have tested it. No matter what you do it is on their lists.
   So the danger is that you create resources with seacrets (user data or sensitive information) Because it's just dev. You compiled it like a dev or test version because you are going to debug it or log something, so it's a bit bulky. Then someone can reverse engineer some files. There might be config files there and some extra files that go through to deployment. By doing this you are really exposing us to this threat.
   And then once the resource is up you end up in the situation that you have an endpoint there, and then you say “well let's go to prod”, then you put protections, you use a domain you own, but in dev and test you still just have the default. subdomain the provider gives you. example : company-resource-type.cloudserviceprovider.io
   Then you just have to rely on that system. But let's say you don’t have any secrets in there. But you were just getting someone to test the resource. And you send a URL to the resource like a webapp or something. And people test it and it's ok. Then there are 2 things that can happen when you delete the resource. And even if you obfuscate the old resource and hide it behind the new endpoint for the real resource, there are people still

with that old link in a chat or email or something. When you delete the resource you don't own that endpoint anymore. Someone else has picked it up and done something with it. IT is no longer your resource. But all the links you sent to that resource still work. Someone will take it, create a honey pot and use that endpoint. And there are 2 things that could happen. You could land on a site that just hacks you. Tries to install something or execute code. But the more likely situation you'll get is a phishing attack. And this happened to us not so long ago. A phishing attack trying to get logon credentials. The simplest thing they can do is create a resource on that endpoint that looks like a login page for a well known service like microsoft or google. And people get this multiple times a day and they think "ah so they have secured the resource with login" and so they type in their credentials. And this happened the other day via internal email that had one of these. So then all of them had to reset all of their passwords and logins. And the capabilities are so high with these hackers that they create these attacks in an hour and it's done.

And even with our team, if it looks like a weird link, they click anyways because people are getting so used to clicking on links. You do it so often that the one that sticks out, you don't really think about it.

And hackers can create these resources really esely. They have full control of the url. They can even use the company name in the url. There is nothing stopping them.

2. **What should have been done?**

   There are 2 things wrong here. Firstly you need to have principles that say that you can NEVER expose web endpoints like this. Always use, and if it's complicated or hard you can automate it or make it a part of the normal routine somehow, always use your own domain. We have our own domain for dev and test. And it's automated, even if you are "just testing something" doesn't matter, it has to go behind the dev or test domain. And it's behind our application firewall and with our domain that we have full control over.

   But what is even better to do. If it's an external facing web-app, something other people outside of our network are accessing, then it has to be very clearly defined how people are authenticated into the resource. If it's just static you should have never done this like this. So ALWAYS HAVE AUTHENTICATION. Even if it's just dev or test.

   So use our domain and certificates and standard authentication. There are many ways to authenticate users. Even if it's just internal, just use our internal authentication. And as soon as you do that it forces everyone to think "what is the minimum security level" and they think it's hard to do every time you want to spin up a resource. SO they automate it. It doesn't matter how you do it. You need to get the infrastructure to be a part of the pipeline. Infrastructure design will be better for it because the minimum requirement will be a part of the default. So set the minimum high.

3. **Who discovered the vulnerability?**

   Noone really noticed. I saw the resource after it was upp. We knew it was a thing that could happen.

4. **Why is it bad? What are the consequences for you or others?**

   Well in this case there was nothing really bad that happened. There was no real data in the resource. It was just dummy data using and the app had no connection to our network. It was talking to an API that only had dummy data. So we did a security check and it revealed that there was nothing compromising with the resource. So we just used it as an example, blew it a little out of proportions so everyone understood that this could

have gone wrong. But now we have to park this resource forever and create a subscription with our cloud provider that is called “DO NOT DELETE”. We can never delete it because that opens up a potential threat.

Why?

1. **Who can do this? What position do you have to have, to be able to do this.**
   We have a structure so that not anyone can do this. Even though you are a dev or someone else on our team only senior infrastructure architects are able to confirm pull requests to even be able to deploy this like this.
2. **Are there any systems in place that could potentially stop these kinds of mistakes.**
   Yes. The problem is that a lot of people have this urge to “it's just a prototype” or say “I’m just gonna test this real quick” and that is a problem when things are supposed to happen fast.
   When we do real product development. We use terraform. A part of that process is when terraform “plans”. There is a check there against policies that we have created. We can control these sorts of things through those. But in theory if you have the rights you can override that.
   But the way to stop this is to say that our policies and requirements are the minimum. If you go below them you are not allowed to deploy anything. Even if it's just dev or test. It has to be automated and put into the pipeline. People should not be doing this. It should be the pipeline. And when something has been misconfigured or is outside of those policies the plan fails, just like if you had a typo somewhere. An integrated part of the system.

3. **What have you done after this incident? What did you and your team learn?**
   We will be enforcing this even harder moving forward. And we are also using our cloud provider to check our logs to see if they find any vulnerabilities.
4. **What security measures are a part of your yearly routine that you think more businesses should do.**
   Code review on pull requests. It’s proactive. It puts pressure on people to do things right. If there is a program checking your code it's no big deal. You get an error. But you are showing a coworker your code you want to do things right. To impress. It’s just how we are. So all preventive measures are key. Things that make people aware of our system and how it works. Pentesting is really just to say that we have done it. (most problems arise from within. So working preventatively within is going to have a better net yield for security.)

   (Infrastructure as code has made it possible to review infrastructure in a much better way)

5. **Is it unreasonable for you to expect everyone on your team to know the systems they work with intimately? Or is it enough that one person knows how it should be and that person takes responsibility for it.**
   It's really important that everyone knows the systems. It's not unreasonable at all unless it is an incredibly large system. Of course you have to have some segregation of modules where certain people know more about certain parts of the system.
   The 2 pizza problem is a good example. If the team on one module is so big you would have to order 2 pizzas for them, then the team is too big. But if you have a product or a module that needs 2 teams you have to behave a bit differently. Then you as the product owner, with 6 programmers or something, then you have to think about infrastructure and make sure that what they are working on can come together. And if what they are working on is supposed to come together into one cohesive product then each team needs to know what the other is doing. Spotify is a good example. They changed their app from one product to different modules for each section of the app. So it can release smaller updates more often. They had to change the fundamental architecture of the whole app.
   It's never enough to have just 1 person one the top that knows the whole structure. Otherwise it will be out of sync.

6. **Do you buy services from 3rd parties? Have you ever experienced vulnerabilities because of 3rd party products or services?**
   Yes we do and yes we have. You better be careful! We are working on a framework to review not only the product itself but also the provider. Just today we had a small confrontation with a provider, not because they did anything, we just don't trust them. And that's really hard. And we have been compromised 1 time that I remember because of a third party. It wasn't that bad, but it could have been much much worse.
   That example was, everything is supposed to have single sign on, and when you buy third party tools for Office or something. And it's some code that you buy that either runs locally or it is talking to a server somewhere (adding a point of failure). And then you have to do a security review. Is there a possibility of some data going out of our system. No, okei then we have to ask 3 more times, are there any possible connections from our network through this? And if it is then we need to know what and get documentation. And because of norwegian law it has to comply with the "Data Behandler avtalen". We need to know if it handles data. Where that data goes, and if it is GDPR data we have to go even deeper.

7. **How can smaller businesses protect with less technical knowhow protect themselves against these types of 3rd party vulnerabilities.**
   It's almost hopeless. Let's say you are a plumbing company. And you use google professional or something. And someone buys extensions for g-suite. How are you supposed to know that that is safe? Well you would have to google it. That's how it is today. How do you know that it's safe to install an app on your phone? Well you just trust Apple, or google. So when you are a business, you just have to pick a provider or service that you trust. And it says it's ok, you just have to trust that judgement.
   Should people choose to just 'upload an excel document' to Joe Blow, who just fixes it. Because he knows the password to the sketchy site we use and we just hope and pray it works

Or should you use some standardised product from a reputable source. Check what other people in your sector are doing. The social network is paramount if you don't have the technical knowhow.
And the mistake a lot of bigger companies do is that they say " we want to hold all doors open so we will use a multi cloud strategy that has all of the services available" but then you need even more capacity for infrastructure and security. And if you don't have it you will end up taking shortcuts.

System specific

1. **Pros and cons of Cloud Services. Security wise.**
   Security is one of the pros, yes. It's more complicated but more flexible. The main pro is agility. To be able to use new things and be able to move them around.
   But the main security pro is the trust factor. Everyone knows who microsoft or amazon is. So do you believe that microsoft or amazon have better security than Joe Blow who has a 'datacenter' in his garage. Yes!
   It's about reputation. And even the government has put rules and laws in place so that user data is not stored in 'non reputable' cloud services. It's mandated.
   Also since we are all stuck at home. Who do you think is better at scaling their infrastructure now that there is more traffic. I am not betting on Joe Blow.
2. **What is your opinion or your team's opinion on implementing features that are safer but might not be well received by end users.**
   It was a theme last year. We were implementing some set of features that really were necessary for the security or our data. MFA and password changes and what not. Of course you could just activate these changes right away if you don't care about the user. But when users do not like something there it might not be a bad feature or implementation they just don't understand the changes. They have not been given the time or information to see why these changes might not be so bad or even improve their workflow. They also possibly don't understand the danger of not having some security features. They don't understand how these features work or why they are important.
   So before we do anything we have to go and educate our users. So we spent a lot of time doing that. Educating our users both on the severity of bad cybersecurity and also what the new features do and how they work and most importantly how they keep them safe. So we train people in thinking about cyber security. And then instead of thinking "oh this will be hard for our users" we spun that around and thought how could we make this easier for our users. What could we do even before any changes are implemented?
   So it's a lot of communication with our users. And then when it comes to managing these changes, we had our top people trained and educated to be an example for everyone else. If they can do it so can you. If your senior officer can do something you should be able to also. So we got them on video saying why this is important, how they can do it and why it is better.But then we have to be sure of what we are doing is not necessary. We could think of it in a way. What gives us the most security whilst causing our users the least amount of trouble. Don't add anything that is not necessary.
   Single sign on is a security measure believe it or not. It is very good for the user. Less distraction and there are policies that are trusted making sure that users are not logged

on in 2 places simultaneously. So as long as you are on a known device and are logging on from known networks the distractions of the security measures should be minimal. So when we managed that. Having the user's identity bound by rules that make sense and are safe. Noone has complained. Their workflow was not impacted in a bad way. This could have been a terrible change if we had only presented that these changes were going into effect without the training and education that we did. People would have thought it was a terrible idea. But the work we put in made that change almost easy. But it was a lot of work beforehand.
So training, management buying and good communication.

3. **You are a pretty big entity in the Norwegian business sector. But what are your opinions on new tech that maybe you are using but maybe smaller businesses are not caught up with yet. Is it viable for them now or is it still a work in progress.**
   I am not so sure when it comes to smaller businesses. So I am not so sure what is available to them on a smaller scale. But anything that takes identity away from just a username and password and makes it a more suitable set of identifiers that are handled by only trusted providers is a step in the right direction.