Les 4

1.1 Yellow team

Mensen die software maken
System architecture → nadenken over het systeem architectuur

Input sanitizion: Vertrouw alle velden die toestaan om iets in te vullen, bestanden kan uploaden, deze gaan we sanitizen voordat we deze gaan gebruiken.
Dont trust the client: Je weet nooit met wie je praat. Client kan kwade bedoelingen hebben maar ook niet, maar kan ook ongewild de boel kapot maken.
Update your shit: Goed up-to-date blijven met de libraries, extensies, … (zie white team)

OWASP foundation kan je vergelijken met de ISO-standaarden.
OWASP is geen standaard maar een set van guidelines, is bijna een standaard geworden.

Proactive controls zijn controls die OWASP aanbieden
Set van guidelines of checklist, kijken om na te denken van hoe denk ik over security voor mijn webapplicatie, wat zijn de zaken die ik mee rekening moet houden?
Houvast zodat je dat niet vergeet.

Als we nadenken over software en moeten we ook nadenken wat de securityrequirements zijn voor dat stukje software. Gaat over risks en threads. Ook concreet labelen wat de securityfunctionaliteiten die we nodig hebben?
Een misuse case is te vergelijken met een user story.
Deze zaken moet je verifiëren, deze moeten ook meetbaar zijn zodat deze opgelost en aangepakt zijn.

Gebruik de technologie die bestaat
Gebruik de technologie die er is, het wiel niet opnieuw uitvinden op vlak van security.
Als je een pipeline aan het opzetten bent, is het handig om de dependency's mee bij te zetten. Bestaan tools om te kijken als andere tools nog up-to-date zijn om dat te kunnen gebruiken en doen ook aan dependency check.
Als je libraries gebruikt, is het ook belangrijk om enkel de require behaviour gebruikt en exposed zodat men de attackfactors (om aan te vallen) laag houd. Enkel functionaliteiten gebruiken die je nodig hebt.

Secure queries: Zorg ervoor dat je de database aanspreekt op een correcte manier. Zorg ervoor dat de vragen die aan de database gesteld wordt dat die correct en veilig zijn. In bobby-tables.com staat beschreven hoe je verschillende technologieën best aanpakt.
Secure configuration: Denk na hoe je best de database gaat configureren. Zoals welke poorten wordt er gebruikt. De configuratie die je voor databases krijgt, de implementatie is niet veilig.
Secure authentication: Wie heeft rechten op de database? Je kan dit met rollen beheren of op gebruikersniveau beheren.
Secure communication: HTTPS zorgt ervoor dat het over een veilige lijn/kanaal gaat.

Encode and escape
Voordat je in een URL iets zet, zorg ervoor dat je de HTML escaped.
Validate all inputs
Als we data doorsturen, zorg ervoor dat het correct encode en decode is voordat we de data gaan gebruiken.
Data pre-processen voordat je deze niet zomaar gebruikt.
Zorg ervoor dat je niet vertrouwd wat binnen komt, sanitized dat. Voor databasequery's gaat dat over geparamiseerde query's.

Implement digital identity
Authentication → bewijzen wie je echt bent
Level 1: password
Level 2: Multifactor authentication

Multifactor → verschillende factoren → vb. ik heb 2 dingen nodig om mij te verifiëren, bewijzen dat ik het ben. Je hebt een wachtwoord en een app die een nieuwe code genereerd.
Level 3: Cryptographic based authenticaion
Aan hardware denken. Denken welk niveau van toepassing is voor jezelf.
Session management
Bij houden wie je bent, bv. cookies. Bv. winkelwagen van een webshop, je houd bij in die winkelwagen wat je wilt kopen.
Moet eindig zijn en we moeten ervoor zorgen dat we geen gevoelige informatie in die cookies steken.
Op tijd een nieuwe cookiesessie aanmaken. Bij elke privilege een nieuwe sessie starten.

Enforce access controls
Wie mag wat doen?
Deny by default: Beter van 0 starten en privileges (rechten) toevoegen dan alles te bouwen en dan rechten af te nemen. Bengin bij niks, ken iets toe.
Don't hardcode rules: Geen rollen toekennen in code, is slecht te onderhouden. Gecentraliseerd houden en niet verspreiden doorheen de applicatie.
Zorg dat alle requests door de accesscontrols gaan → telkens opnieuwe controles doen zoals: zijn er nieuwe tokens gebruikt, heeft de gebruiker wel de juiste rol?
Protect data everywhere
Mobiele applicaties kan je niet vertrouwen dat die secrets en keys bijhoudt, geen toegang tot secure storage (beveiligde opslag), zeker als die in de code staan, applicaties makkelijk decompilen.

AWS KMS → managed service
Hashicorp vault → open source project dat hetzelfde wil bereiken namelijk een veilige plaats waar je je credentials kunt opslaan.
Qualys → vooral voor validatie, checken van SSL certificaat
SSL Labs → checken van SSL certificaat

Implement security logging and monitoring

Handle all errors and exceptions
Chaos Monkey → randable service uitzetten, zet een aantal services in het systeem soms uit om te zien dat alles robuust is ontwikkeld is geweest dat het systeem om kan als bepaalde services uitvallen.
Gebruik standaarden en verzin zelf geen error codes. Als je exceptions gooit, geen gevoelige data in de exceptions laten staan of toevoegen. Error geeft veel informatie voor een aanvaller.

Les 5

1.2 Yellow team

Vroeger hadden we FATClients maar ook ableservices die verantwoordelijkheden op zich namen maar nu maken we gebruik van services van derden.
Aan de database is een API gelinkt zodat andere software gebruik kunnen maken van die database via de API om bepaalde foto's van koeien op te vragen. We hebben dan de client die intrigeert met de browser die dan de API aanspreekt.
De communicatie tussen browser en API moet over een veilige communicatie gaan, over HTTPS. Als we niet over HTTPS gaan communiceren, werkt oAuth niet.

We hebben cowmatch en bied ook een API aan en we hebben een browser die daar gebruik van maakt. De API van cowmatch wil ook gebruik maken van de API van koebeesten.com. ze bieden een eigen service aan en wil onderliggende gebruik maken van third-party service van koebeesten.com. Als de client gebruik wil maken van de API van cowmatch, moet de client credentials gaan gebruiken om te authoriseren bij de API van koebeesten.com om aan de foto's te kunnen.

Cowmatch kan ook een mobiele client hebben waarbij de werking anders is dan de webapplicatie.

Kan ook zijn dat er een derde partij gebruik wil maken van de koebeesten.com foto's in een andere mobiele applicatie.

Complex ecosysteem van allerlei spelers die toegang wil hebben tot data maar de data is van iemand.

Machines communiceren met andere machines. Bv. services die met andere services communiceren en de ene service wilt aan de foto's van koebeesten.com omdat deze toegang heeft maar de andere services zegt dat hij zich moet bewijzen dat hij toegang heeft tot die foto's.
We willen niet dat machines ons username en password gebruiken om aan authenticatie te doen en wanneer je inlogt dat die niet je credentials gaat gebruiken voor koebeesten.com om zaken op te vragen.

Er steken al lang geen credentials meer in cookies. Je kan geen waardes in je browser steken. Cookies worden gebruikt om bijvoorbeeld een state te onthouden van een website. Je kan geen state onthouden die met security te maken heeft via cookies.

API key worden nog steeds veel gebruikt. Als je data wilt van Google Maps, kan je je applicatie registreren op Google Maps. Google Maps weet dat je bestaat en dan krijg je een API key, met deze API key kan je requesten sturen naar Google Maps. Er is een probleem met de client in verband met de API key, als je de API key in frontend zet, kan de gebruiker deze inspecteren op de browser. Wanneer je de frontchannels passert (bv. browser, mobile client) kunnen we niet meer met de API key werken.

Single Sign On (SSO)
Een valid key is een token en is een bare-token. Een bare-token is een token waar andere mensen dezelfde rechten heeft. Kijkt niet wie het is.
De bare-token moet goed beschermd worden.

Als je bijvoorbeeld een account hebt bij koebeesten.com en heb een paar foto's op de database staan van koebeesten.com. in dit geval ben je de Resource Owner. Resource Owner is de eigenaar van de resources die hij heeft. Resource Provider is diegene die de resources geeft.

oAuth is een service die ook kan draaien op dezelfde services zoals anderen. Er is een andere service en noemt Security token service, is degene die tokens gaat uitgeven.
Oauth werkt op dezelfde manier als een valid key. Wanneer cowmatch foto's wil halen bij koebeesten.com geef men de sleutel, geen username of password en geeft men aan koebeesten.com.
Security token service is de authoriteit van de sleutels die tokens uitkeert. Alle autorisatie komt op één plaats terecht.
Voordeel: alle authorisatie komt op een centrale plaats terecht, er moet dan op 1 plaats onderhoud gebeuren. Als we updates hebben, moet deze op 1 plaats aanpassen.
Nadeel: Single Point of failure, als de service gecompromised is, hebben we een groot probleem.

Oauth heeft flows, zijn scenario's van hoe ik moet omgaan met oauth en welke oplossing is correct voor de situatie.
Client credentials
Machine naar machine communicatie → een service die aan de andere service iets wil doen. Als je de API key hebt, zou je rechtstreeks naar de Resource Provider moeten communiceren en moet normaal gezien de foto's van de koeien geven.
Bij oauth is er een derde partij, namelijk de Security token service. Elke API die ingeschreven is bij de security token service. Je krijgt eerst een client id en een client secret toegekend, alleen de client secret mag niet publiek beschikbaar zijn. Als je de foto's wil hebben van koebeesten.com, de security token service moet eerst antwoorden met een access key. De resource provider moet connectie leggen met de security token service en vragen als de security token service een token heeft uitgegeven. De security token service gaat antwoorden met ja, dat was ik. Deze controle vind plaats achterliggend bij koebeesten.com. hier is geen usre interactie aan toegevoegd.

Authorization code flow
Men gaat via de browser naar de website van cowmatch. Men wilt met cowmatch integreren voor de eerste keer. Cowmatch zegt wil je connecteren met de koebeesten.com-account. Met het cowmatch account kan men ook foto's van koebeestn.com binnenhalen en aan cowmatch toevoegen. Om dat te doen moet er eerst toestemming gevraagd worden. Request gaat van de API naar de browser en gaat een request sturen naar de

security token service en zegt dat men is gestuurd door cowmatch, kan men ervoor zorgen dat het cowmatch account aan het koebeesten.com is gelinkt? Je krijgt dan toestemming om zeker te zijn dat jij het effectief bent. Als je toestemming hebt gevraagd en gekregen hebt, wordt het geredirect met een access code naar de browser. De browser gebruikt de redirect en access code naar de API. De access code vervalt na een tijd. Redirect is dat de API weet met welke security token hij moet praten.
Met de redirect en de access code gaat de API zelf connectie maken met de security token service en krijgt een access token terug. De access token wordt gebruikt voor de resource provider die dan de access token gaat valideren bij de security token service en men krijgt de koeienfoto's terug.
Wat links staat (resource owner en browser) is frontchannel en onveilig. De access token passeert nooit langs de browser. Alleen de access code passeert de browser. Wanneer de access code gebruikt wordt, wordt die onklaar gemaakt zodat niemand anders deze kan stelen zodat er 2 access tokens bestaan. Het redirect moet ook kloppen. Als de request niet via de API is gepasseerd, negeert de security token service de request.

JWT token is een bare-token. Heel belangrijk dat de bare-token niet gestolen wordt. Als het niet werkt met HTTPS en als er een Man In The Middle is, kan de token gestolen worden en hergebruikt worden, er wordt geen controle gedaan van waar of van wie de request komt.
Als de tokens vervallen, moet men de Authorization code flow helemaal terug opnieuw doen.

Bij de security token service moet de API geregistreerd zijn en moet een legitimate redirect URI zijn. Wanneer men van de API naar de service token service een paje vraagt, moet niet alleen de client id en client sectret kloppen maar ook de URL moet gekend zijn bij de security token service zodat we niet via een domein een request krijgen dat we niet vertrouwen, zo een pasje aanvragen gebeurt via de backchannel-request, gebeurt nooit via browser.

Les 6

1.3 Yellow team

Iedereen communiceert met iedereen.

Authorisatie goed doen is heel belangrijk wanneer je een applicatie bouwt.
Public security token service zijn extern gehost en je gebruikt die zoals welke API's mogen communiceren, welke redirect URI's vertrouw ik? Je hebt daar volledige controle over.

Ze doen suggesties hoe de access token eruit moet zien.
Bearer token: Wie de token heeft, is de eigenaar van de token.

Bij puntje 8 heeft API een access code gekregen en dit ben ik. Moet zich identificeren als cowmatch.com en heb de access code en dat steunt op een client secret dat gedeeld is tussen de API en de security token service.
API is een instantie die een secret kan bijhouden. Mobile en Single-Pages kunnen dat niet.
Een signature is een public-private key en staat op de token.
Op een access token staat een refresh token en de API kan als de access token vervallen is met de refresh token naar de security token service gaan en vraagt voor verlenging van de token, de security token service mag die verlenging geven. De refresh tokens kan je maar 1 keer gebruiken. De security token service bepaald wanneer de token niet meer verlengd mag worden. Als de gebruiker geen toegang meer heeft, kan die met de refresh token geen nieuwe access token meer krijgen.

Security token service geeft de legitimate redirect URI naar de browser en de browser stuurt deze door naar de API. Legitimate redirect URI zijn ook niet te vertrouwen in mobiles omdat wanneer ze een redirect binnen krijgen is dat een system-white-call waarop iedereen zich kan inschrijven. De legitimate redirect URI moet geregistreerd worden bij de security token service.

Als de flow slaagt en krijgen een access token, dan heeft de gebruiker zich geauthentiseerd.

Openid connect
Uitbreiding op het oauth principe. Heel het proces komt terug aan bod maar je krijgt nog een access-sleutel bij met bepaalde scope en met die access-sleutel kunnen ze naar de resource provider gaan en laat weten wie het effectief is. Men kan alleen een deel openen en laat via daar weten dat het die persoon effectief is.

Het is niet dat je aan de foto's van koebeesten.com aan kan als je ingelogd bent. Is 1 proces, hier wordt de access token en de sleutel meegegeven aan de security token service. De flow loopt 1 keer.

Proof Key for Code Exchange
Uitbreiding op de flow.
Normaal heb je een client secret nodig om te kunnen communiceren met de security token service. De security token service weet dat je bent wie je beweert wie je bent.
In mobile genereren ze een code verifier en een code challenge. De code challenge is een hash van de verifier. Een hash is one-way. De code challenge wordt meegestuurd inde front challenge request, als we terug uitkomen bij de API en wisselt de access code in door de access token, dan stuurt die een code verifier mee. De security token service doet de verificatie. Het enige wat die moet doen is een random string genereren.

Single page applications
Er was een oplossing voor die legaal is en hadden een ander soort flow en staat bekend als de impliciete flow, deze is niet veilig.
Nadeel: geen data bijhouden.
Authorization code flow is een protocol volgen zodat machines communiceren met elkaar in naam van een gebruiker en de gebruiker wordt getrokken in het verhaal, om die machine in gang te zetten moet de machine eerst zeggen dat men gaat communiceren met iemand die iets van je weet. Passeert de security token service die het gaat managen.

SAML
Is hetzelfde als single sign on.

Spreken van een identity provider en een service provider. Wanneer de client iets vraagt aan de service provider, gaat de service provider aan de identity provider vragen, als het ja is, gebruiken die ook een access token, geen JWT tokens maar wel in XML. XML is kwetsbaar voor toestanden als het niet goed gepatched is.

SAML examples steps
Je probeert een webpagina te bereiken, je hebt een serviceprovider die kijkt wie je bent en checkt dat met de identityprovider (ongeveer hetzelfde als security token service) als je die machtigen mag hebben en keert een SAML-token uit.

Bij SAML wordt er regelmatig security bugs ontdekt maar meer bij implementaties, niet in specificaties.