Rob van der Veer - Software Improvement Group
@robvanderveer
r.vanderveer@sig.eu

SIG ziet elk jaar de broncode van honderden systemen en ik wil u graag meenemen op een tour langs wat wij tegenkomen als het gaat om security en privacy by design in de software die wij onderzoeken.
Uiteraard kan ik de details van onze klanten niet met u delen, vandaar dat ik wil beginnen met een illustratieve en pijnlijke case uit de actualiteit.

Veilige software lukt nog niet. Wat nu?

Dating-site Ashley Madison (ziet u de keurmerken?) werd gehackt door een te eenvoudig VPN wachtwoord: Pass1234, dus security was niet goed, maar daarnaast werd er ook niet goed omgegaan met persoonsgegevens. Adresgegevens en ip-adressen werden onnodig bewaard. Die werden na de lek allemaal gepubliceerd op het internet. Zo konden werkgevers bijvoorbeeld zien wie vanaf werk gebruik had gemaakt van deze site.

Een ander voorbeeld is de app Runkeeper van Nike die de gps lokatie van gebruikers doorspeelde naar diverse advertentiebedrijven, ook als je niet aan het hardlopen was

We zien steeds dezelfde securityfouten

Inconsistente autorisatiechecks

Zelfgemaakte cryptografie

Verkeerde wachtwoordopslag

Onvoldoende bescherming tegen SQL injectie en XSS, gekopieerd van StackOverflow

“Oh ja, dat moeten we nog uitzetten”

Certificaten wel opvragen maar niet controleren

Onvoldoende logging en bescherming daarvan

Onversleutelde interne communicatie met wachtwoorden etc.

Alle collega’s mogen gegevens zien

Achterdeur voor onwikkelaars

Hoe staat het ervoor met security by design in de praktijk? Welnu, het CPB spreekt in een recent rapport van ‘Marktfalen’ en dat is ook wat we dagelijks in de kant kunnen lezen en wat wij zien als we systemen onderzoeken

We zien fouten waar je op kunt wachten

We zien ook dit: moeilijk aan te passen software waar het wachten is op een nieuwe kwetsbaarheid of een datalek. Daar hoeft dan in principe geen hacker aan te pas te komen: de developer wordt de dreiging.
De onderhoudbaarheid van software kan dus ook een securityrisico zijn en het komt regelmatig voor dat we adviseren een systeemonderdeel op te schonen of zelfs opnieuw te ontwikkelen.

We zien ook steeds dezelfde privacyfouten

Onnodig gegevens verzamelen

Gegevens niet echt verwijderen

Niet aggregeren

Gegevens te lang bewaren

Persoonsgegevens centraliseren

Overanonimiseren

Externe partijen vertrouwen

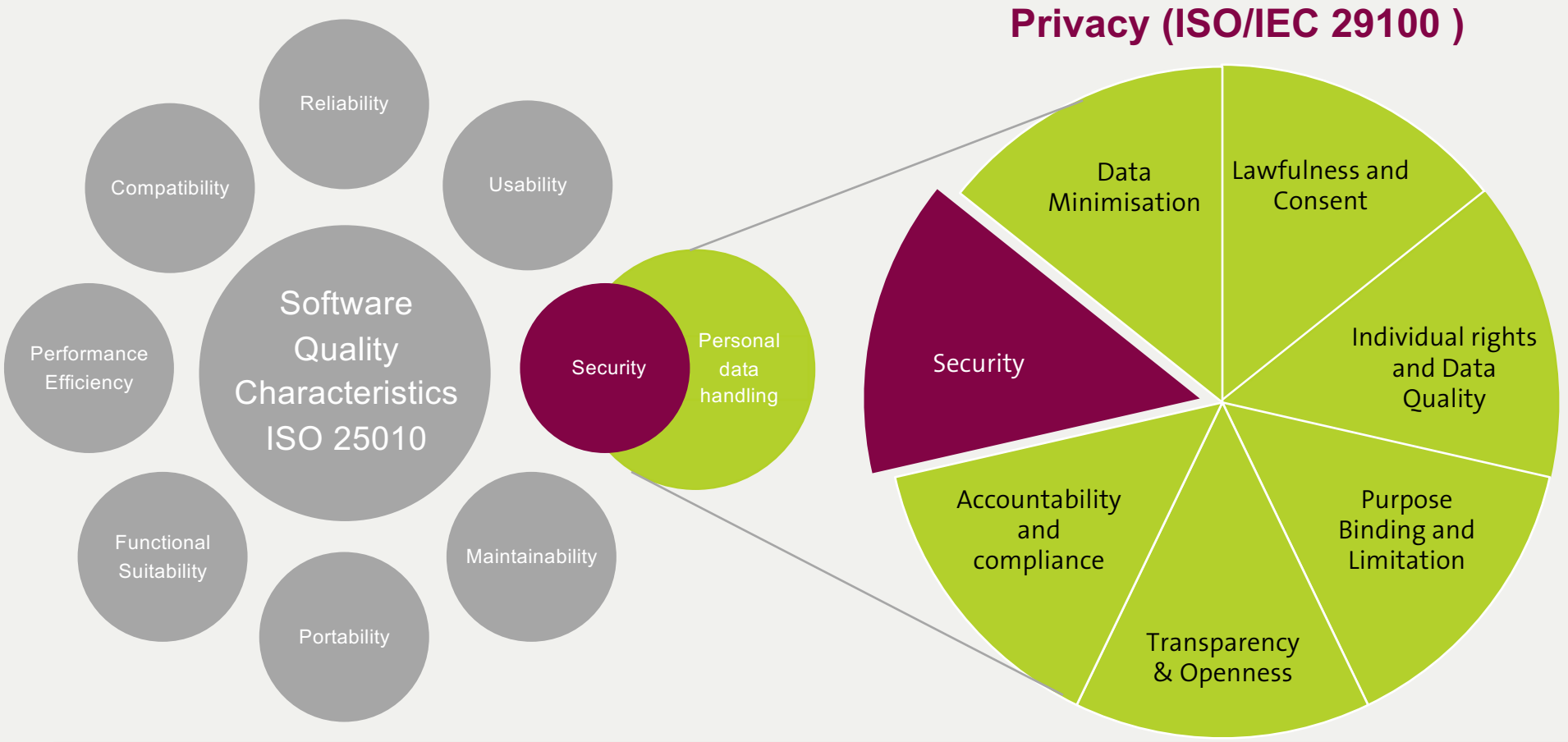
Persoonsgegevens op veel plekken gebruiken, zonder link

Onderschatten van anonimiseren

Gevoelige gegevens in debuginfo

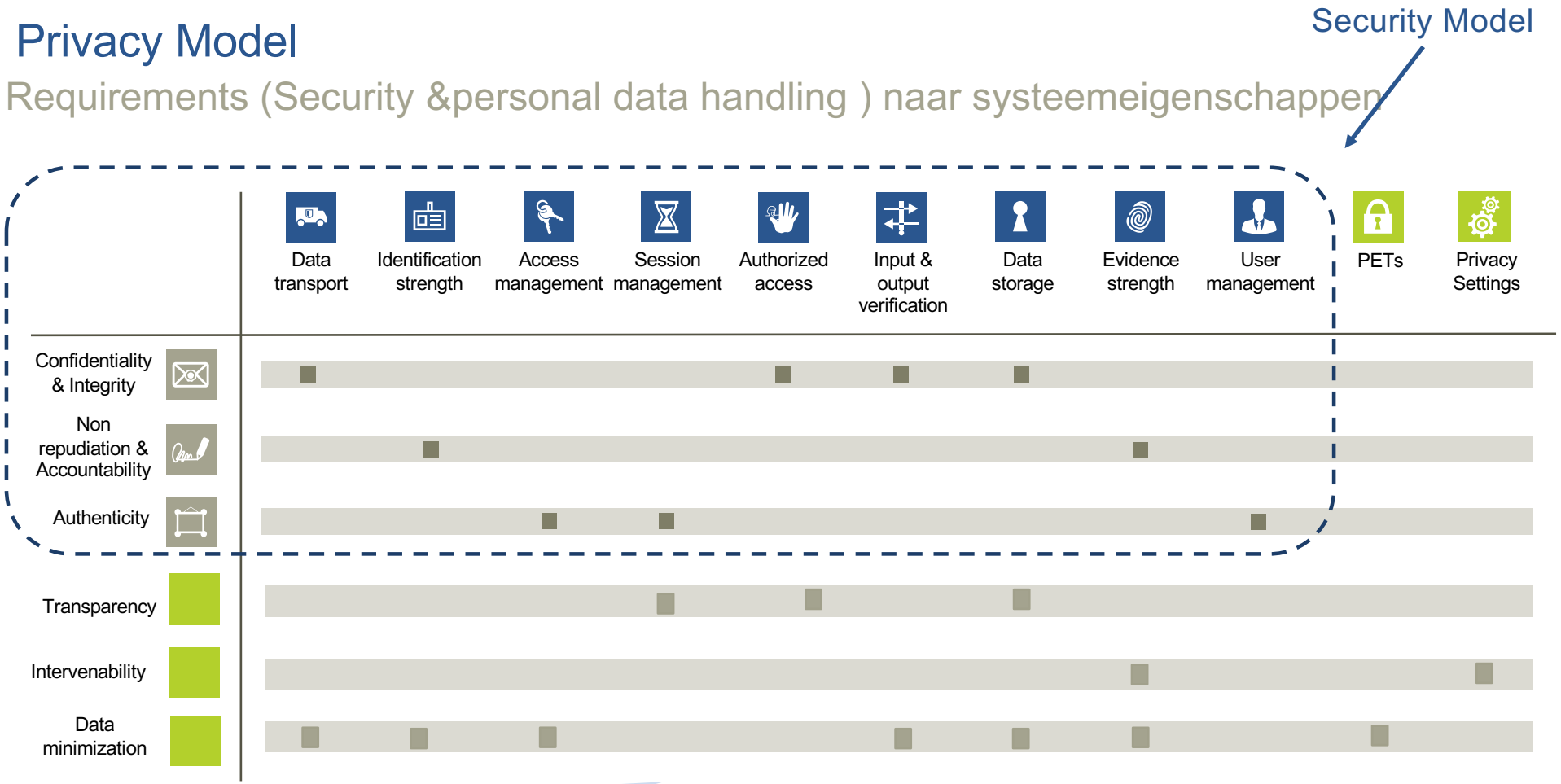
Een derde categorie van issues heeft te maken met privacy, maar het is geen security: onnodig veel risico creëren op het gebied van persoonlijke gegevens. Er zijn vaak goede redenen voor, zoals software-engineering principes en business intelligence, maar wat we veel zien ontbreken is een goede afweging. Nu volgt een toelichting op de relatie tussen security en privacy zoals wij die zien, op basis van ISO25010 en ISO 29100.

Softwarekwaliteit (ISO 25010) & Privacy (ISO 29100)



Privacy Model

Requirements (Security &personal data handling) naar systeemeigenschappen



Dit is het privacymodel waarmee SIG software onderzoekt op privacy by design en waarmee we dat ook kunnen scoren op een schaal van 1 tot 5 sterren. Duidelijk is te zien dat ons securitymodel hier onderdeel van uitmaakt.

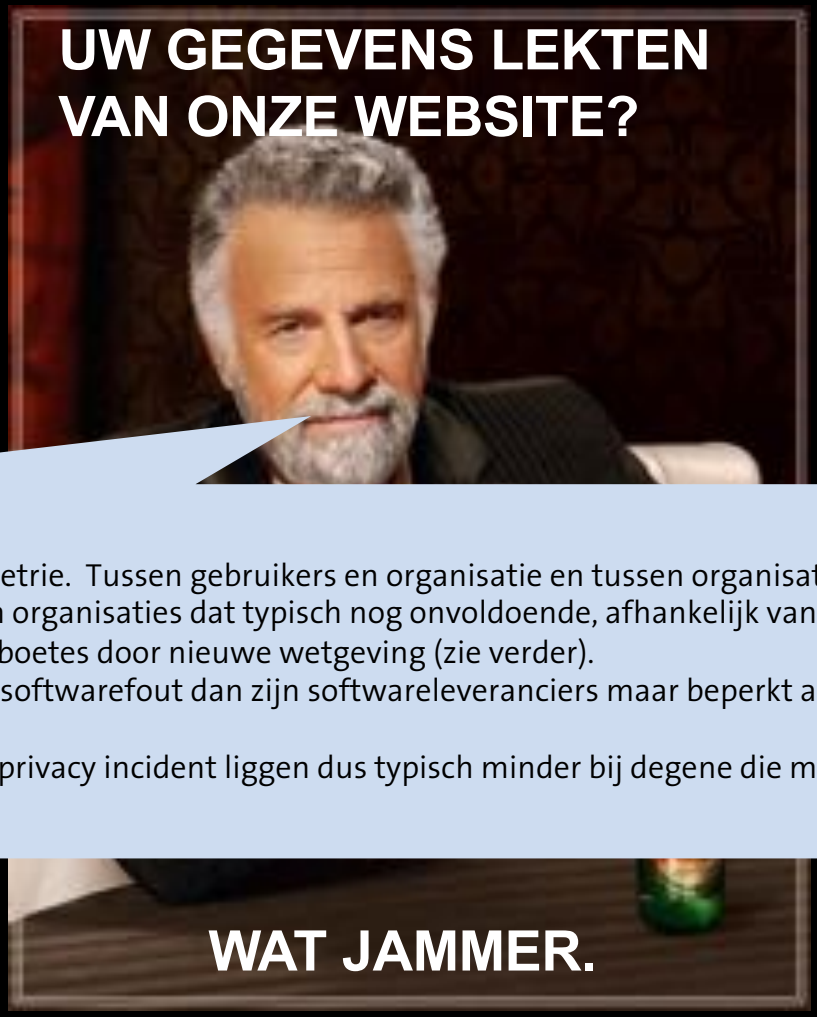
Waarom is het zo matig gesteld met security en privacy by design? Ik onderscheid drie hoofdredenen.
Als eerste: onmatigheid: we hebben weinig tijd en weinig aandacht voor security en privacy omdat software-ontwikkelaars veel te druk zijn met de grote hoeveelheid software van matige kwaliteit.

We maken meer software dan we aankunnen:
12 miljoen software ontwikkelaars
120 miljard regels code in een jaar
15% daarvan moet het jaar daarna worden aangepast
Met 1,8 miljoen nieuwe ontwikkelaars

Gevolg: alleen reactief onderhoud, kwaliteit lijdt en dat maakt het probleem alleen groter, systemen lopen vast of lekken data

De achterliggende oorzaak is dat innovatiedrang niet beteugeld wordt omdat de problemen van te veel software pas later echt een probleem worden. Software bouwen is technische schuld creëren.

1. We maken meer software dan we aankunnen



De tweede hoofdreden is schade-asymmetrie. Tussen gebruikers en organisatie en tussen organisatie en leverancier.
Als persoonsgegevens lekken, dan voelen organisaties dat typisch nog onvoldoende, afhankelijk van het type organisatie. Er kan wel sprake zijn van ernstige imagoschade of boetes door nieuwe wetgeving (zie verder).
Als een organisatie schade lijdt door een softwarefout dan zijn softwareleveranciers maar beperkt aansprakelijk.

De negatieve gevolgen van een security/privacy incident liggen dus typisch minder bij degene die moet zorgen voor veilige software.

2. We hebben niet dezelfde belangen

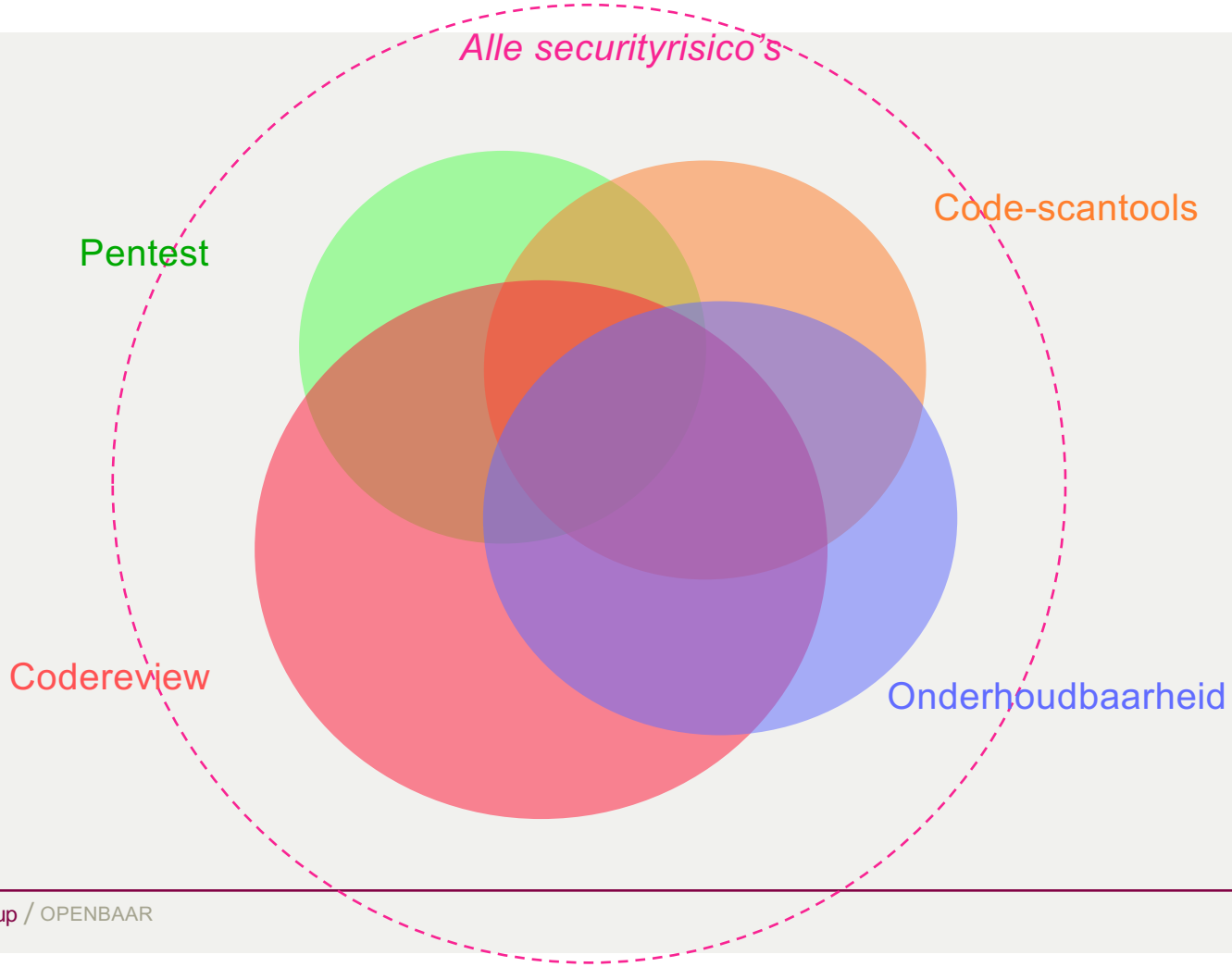
De derde hoofdreden is dat het nog niet zo goed lukt om veilige software te coördineren: interne en externe aansturing van het ontwikkelproces, klant/leverancier dialoog, toetsing, etc.
Waarom?

1. Opdrachtgevers verwachten kwaliteit, maar ze weten niet hoe ze daar specifiek in moeten zijn. "Wij nemen aan dat onze leveranciers veilige software maken". Maar wat vind jij veilig en wat niet? En hoe ga je dat toetsen? Het landschap van standaarden is uiterst gefragmenteerd
2. Leveranciers (intern of extern) beginnen er typisch niet over. Het is niet in hun belang. Misschien is het wel heel lastig om aan de eisen te voldoen. Eigenlijk zouden ze moeten zeggen "Sorry, maar ik kan je geen garanties geven over kwaliteit als je niet specifiek bent". Daarmee zijn ze minder interessant voor opdrachtgevers dan bouwers die dat niet zeggen. Dit is een prisoner's dilemma. Zie verder voor een idee om dit vanuit de software-ontwikkelaar toch in gang te zetten.

3. Coördinatie veilige software gaat matig

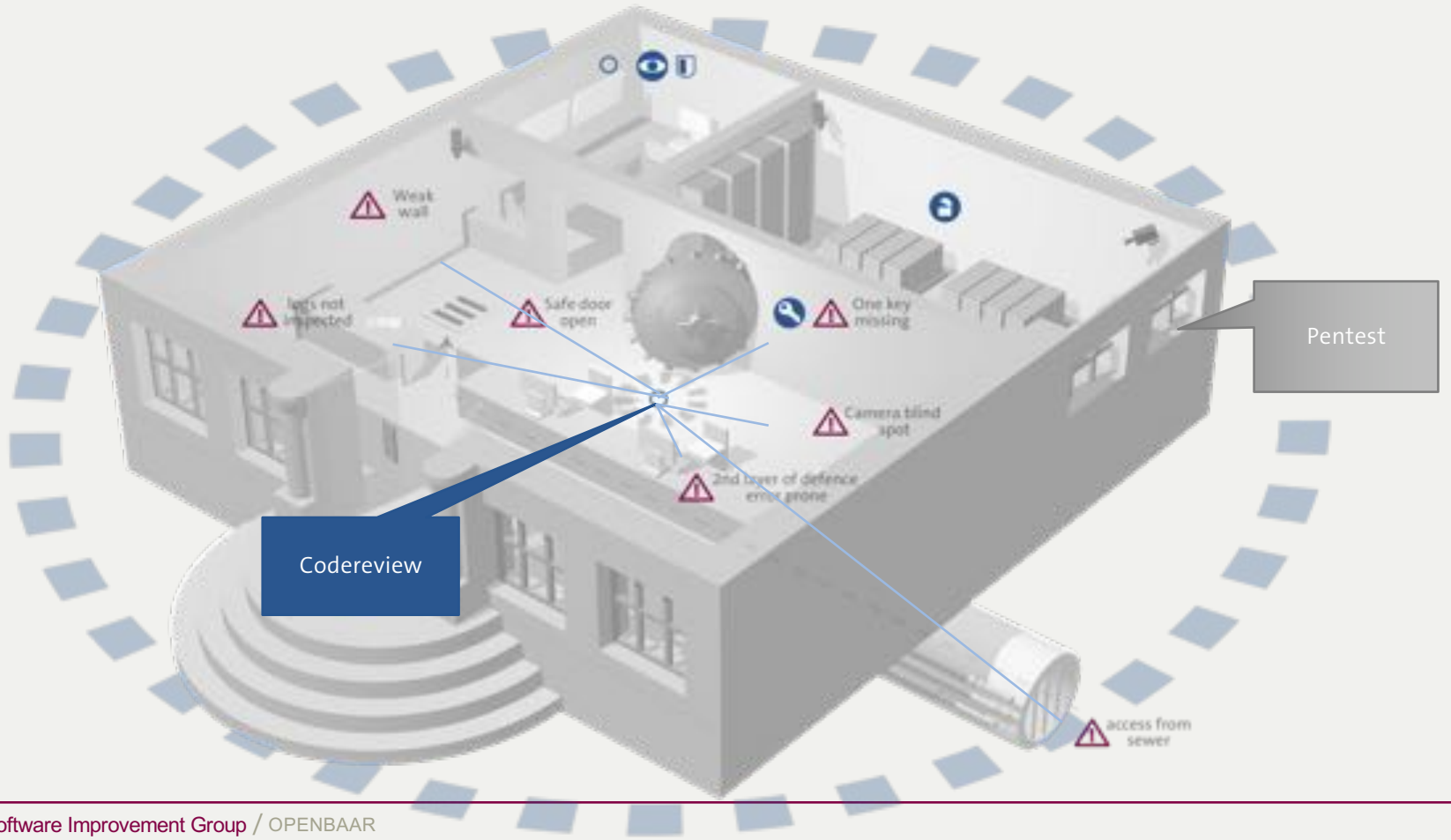
Om één of andere reden wordt er weinig naar code gekeken, terwijl het probleem daar zit.

Hoe toetsen van software security



Codereview vs. Penetratietest

Kijken hoe security is ingebouwd vs. Proberen in te breken



Kwaliteit is dus nog mosterd na de maaltijd. We maken veel en matige software die we onvoldoende kunnen onderhouden en we verzamelen meer data dan we kunnen beschermen.
Is het glas nou halfvol of halfleeg?
Ik zie het glas halfvol.
Ja we worden steeds afhankelijker van IT, maar we hebben dat ook steeds meer door.
Er worden echt wel goede stappen gezet, zoals bijvoorbeeld Grip op SSD.

Wat nu?

- Maatregelen die de pijn leggen bij de verantwoordelijke: zie AVG/GDPR
- Keep it simple en maak onderhoudbare code
- Borg de dialoog over kwaliteit, privacy en security
- Zorg voor meer duidelijkheid in de markt over hoe dan: zie Grip op SSD
- Als de eisen er zijn, volgt de rest:
  - Volwassener worden ontwikkelorganisatie: zie MS-SDL
  - Beter informeren van ontwikkelaars
  - Vroege en brede toetsing inclusief code review (tools, peers, specialisten)
- Bouw security en privacy technologie in principe niet zelf

Grip op SSD practitioner community

centrum informatiebeveiliging en privacybescherming

- 30 organisaties (overheid, system integrators, experts)
- Delen ervaring en doen de standaard groeien
- Nieuwsbrief, bijeenkomsten, werkgroepen
- Links met OWASP, NCSC, ENISA
- Zie www.gripopssd.org

Ministerie van Binnenlandse Zaken en Koninkrijksrelaties

Ministerie van Volksgezondheid, Welzijn en Sport

Ministerie van Financiën

Dienst ICT Uitvoering
Ministerie van Economische Zaken

Nationaal Cyber Security Centrum
Ministerie van Veiligheid en Justitie

Agentschap NL
Ministerie van Economische Zaken, Landbouw en Innovatie

Belastingdienst

SVB

UWV

IBM

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Software Improvement Group

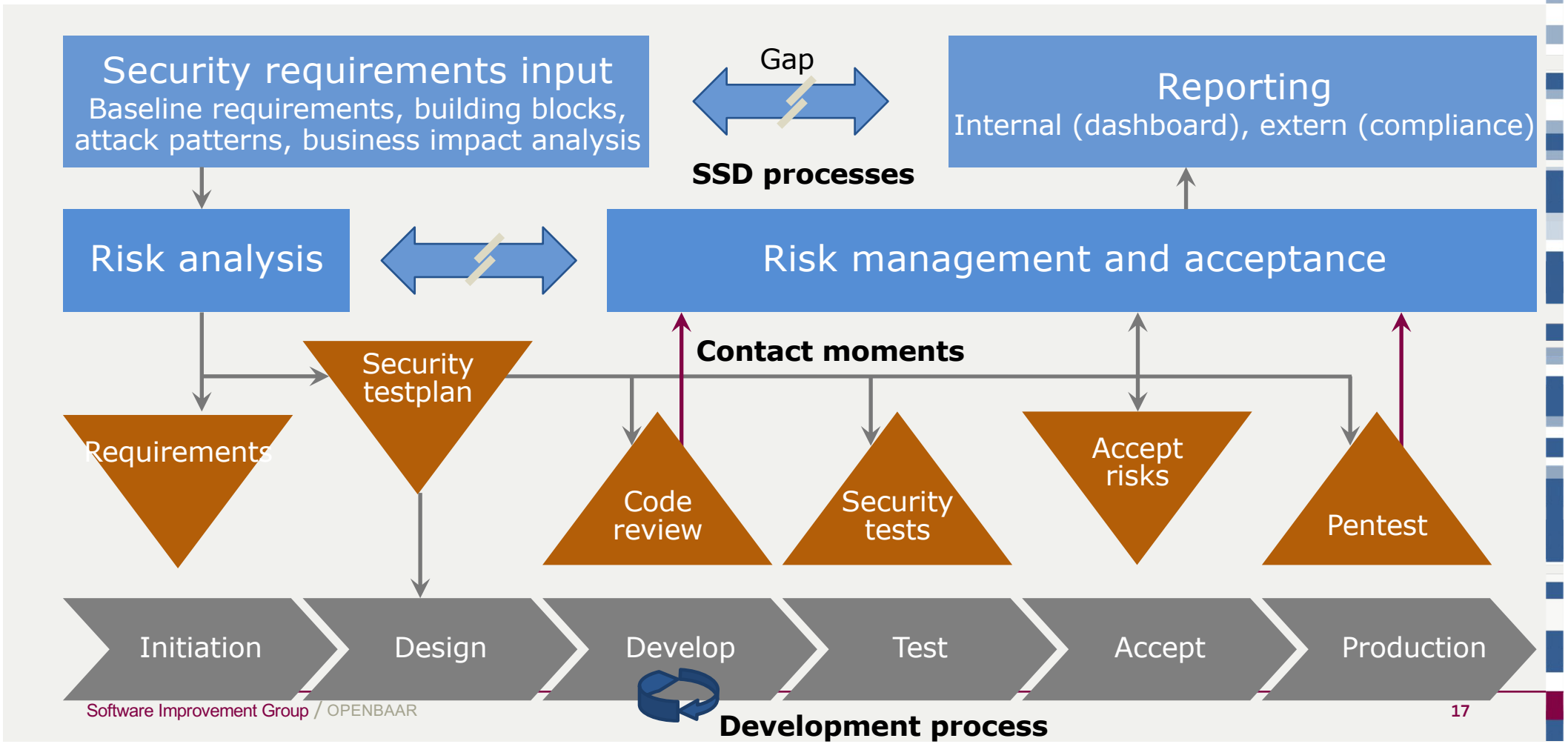
ORDINA
CONSULTING | ICT | OUTSOURCING

SOGETI

centric

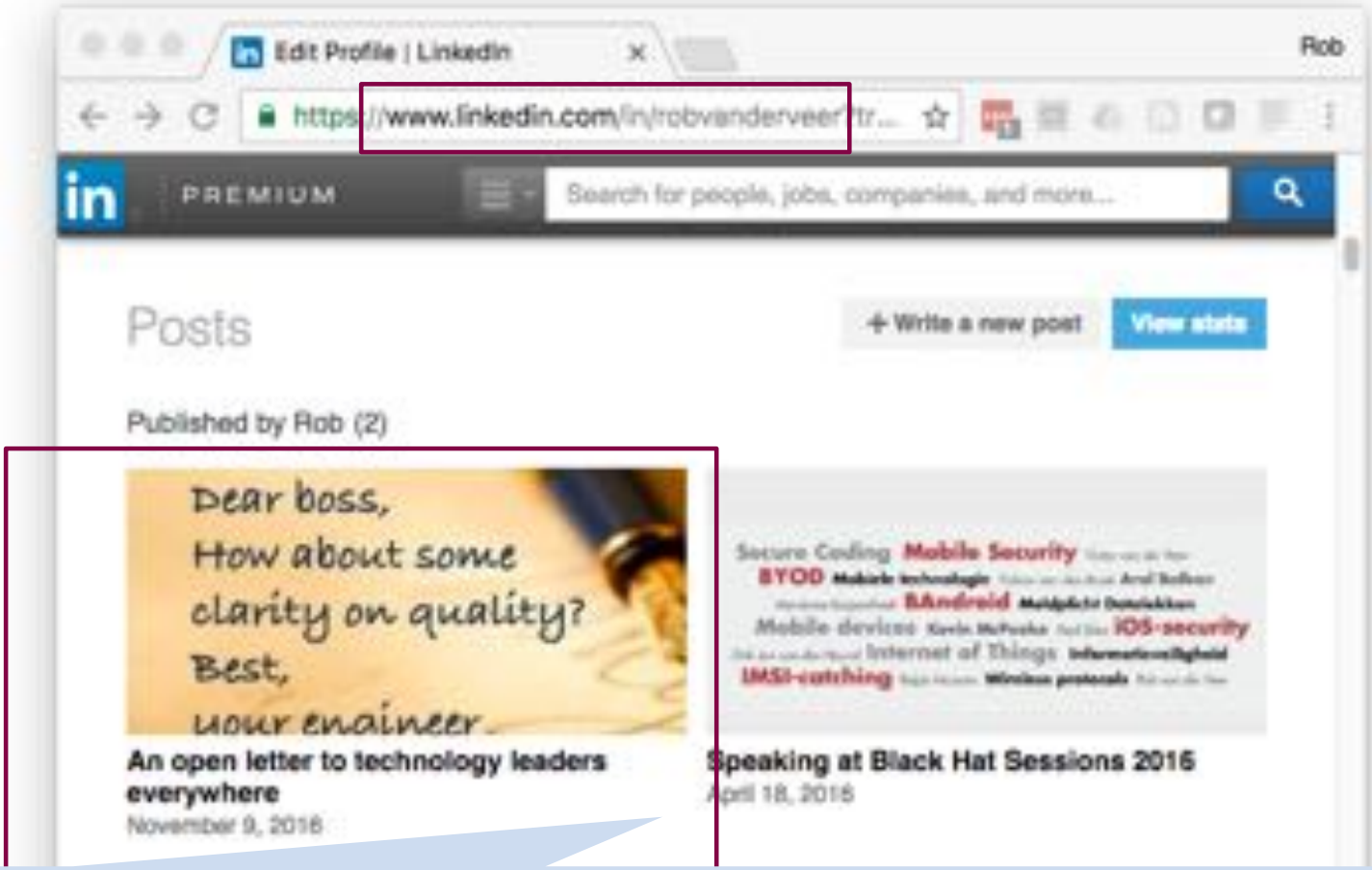
CGI

Grip op SSD


Security requirements input
Baseline requirements, building blocks, attack patterns, business impact analysis
Gap
SSD processes
Reporting
Internal (dashboard), extern (compliance)
Risk analysis
Risk management and acceptance
Contact moments
Security testplan
Requirements
Code review
Security tests
Accept risks
Pentest
Initiation
Design
Develop
Test
Accept
Production
Development process

Hanteerbare richtlijnen voor ontwikkelaars

- Betrek ontwikkelaars in dreigingsanalyse
- Focus op basisprincipes van security en privacy
  - Bijvoorbeeld: het zien van persoonsgegevens als gevaarlijke stof
- Werk met coding guidelines die trigger gebaseerd zijn (zie SIG guidance for producers)
- Informeer ontwikkelaars over technologie en patronen voor security en privacy



Zoals gezegd is het voor softwarebouwers typisch niet uitnodigend om te zeggen: Je moet met mij afspraken maken anders kan ik niet garanderen dat het goed in elkaar is gezet.
Maar het is wel in het belang van security en privacy. Het is ook de manier om goede softwaremensen aan te trekken en te houden. Daarom heb ik het idee opgevat ontwikkelaars een handreiking te doen om een brief te sturen naar hun baas hierover.
Die brief is te vinden op linkedin, op mijn profiel. Zie de URL. Neem eens een kijkje en share of like het als je enthousiast bent.
Mail mij voor de Nederlandse versie of als je wil delen hoe het is gegaan met de brief. Ik ben zeer geïnteresseerd. Succes!

Software Improvement Group

Contact

+31 6 20437187

r.vanderveer@sig.eu

@robvanderveer @sig_eu