The background of the slide is a dark blue, textured surface. In the upper left, there is a faint, glowing globe. Overlaid on the globe is a white padlock icon. To the left of the padlock, the text 'https://www' is visible in a light blue, semi-transparent font. The overall theme is digital security and web application safety.

Securing the code and waiting for  
skilled hackers



**OWASP**

The Open Web Application Security Project



- >15 years information security

Age is not only a disadvantage!

- Security engineer for Defense Intelligence
- C(I)SO for Telco, Retail, Banking
- Trainer (CompTIA Security+, CISSP)
- Security advisor / Technology Architect
- Secure the future:
  - Pushing my little girl toward black belt

# Real Attack



# OWASP

The Open Web Application Security Project

**ABC NEWS**

**SET LOCATION**  
for local news & weather

[Home](#) [Just In](#) [Australia](#) [World](#) [Business](#) [Sport](#) [Science](#) [Arts](#) [Analysis](#) [Fact Check](#) [Programs](#) [More](#)



Source: <http://www.abc.net.au/news/2015-07-20/mick-fanning-attacked-by-shark/6633520>

[Print](#) [Email](#) [Facebook](#) [Twitter](#) [More](#)

## Mick Fanning attacked by shark

### TOP STORIES

- Political stuntman Nick Xenophon gambles on another career flip



## OWASP

The Open Web Application Security Project

### ESPN's Documentary On 'Shark Attack' Surfer Mick Fanning Is Brilliant [Video]

05 Aug 2016 by [Sloane Hunter](#) in [Lifestyle](#), [Sport](#), [Surfing](#), [Video](#)



Source: <http://www.2oceansvibe.com/2016/08/05/espns-documentary-on-shark-attack-surfer-mick-fanning-is-brilliant-video/>



Limited to pen-testing team's skills  
Exploiting several vulnerabilities  
Harmless and trusted  
Good exercising  
Time restricted  
Localized



Swim and have fun!



**OWASP**

The Open Web Application Security Project

This is what you know from previous breaches.  
Einstein was right: time & space are relative.

Exploit everything

Not a CTF!

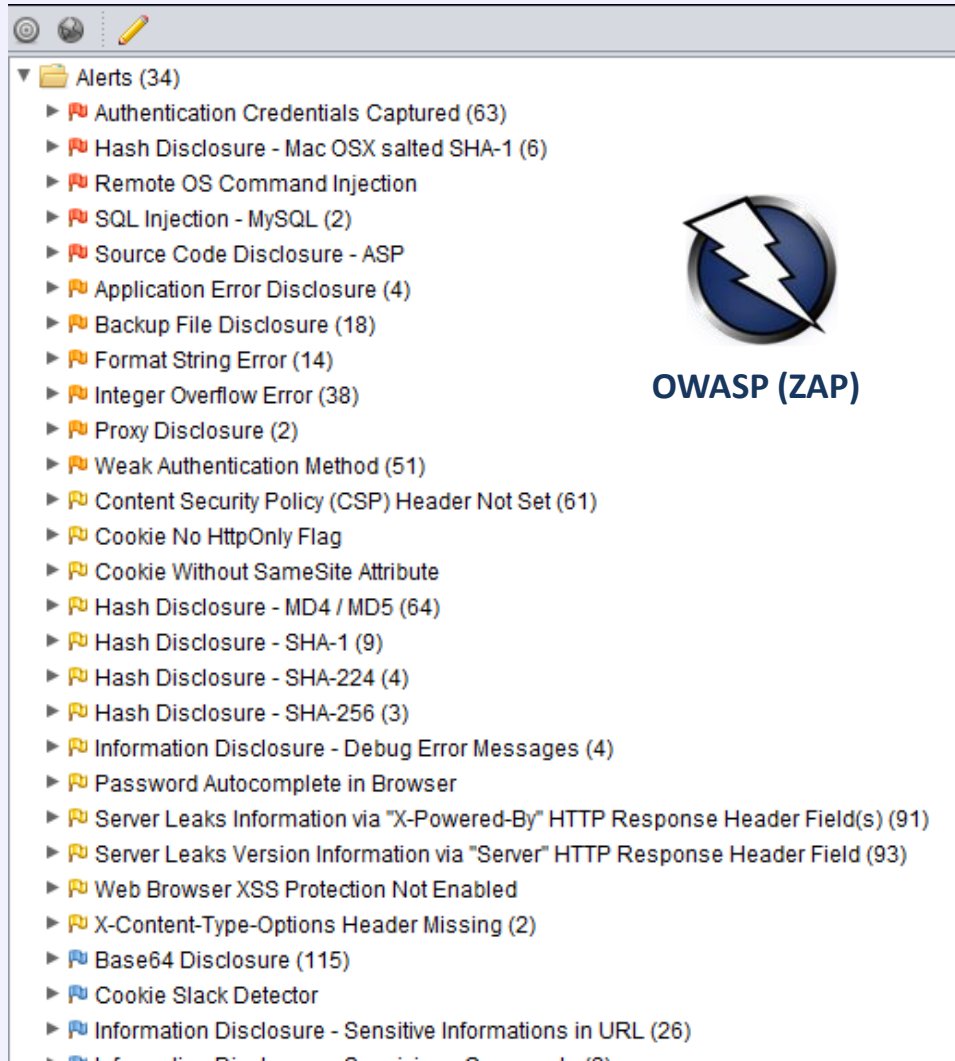

Harmful

A photograph of a swimmer in blue water. The swimmer is wearing a blue cap and a blue shirt with the number '7' on the back. A shark fin is visible in the water to the right of the swimmer, creating a splash. The water is a deep blue color.

Swim if you can!



- Some vulnerabilities exploited during the 7 days exercise:
  - SQL Injection
  - Remote OS Cmd Injection
- More than 95% of automated scan findings - not exploited
- All results should be reported

A screenshot of the OWASP ZAP Alerts window. The window title is "Alerts (34)". It displays a list of alerts with expandable arrows and counts. The alerts include: Authentication Credentials Captured (63), Hash Disclosure - Mac OSX salted SHA-1 (6), Remote OS Command Injection, SQL Injection - MySQL (2), Source Code Disclosure - ASP, Application Error Disclosure (4), Backup File Disclosure (18), Format String Error (14), Integer Overflow Error (38), Proxy Disclosure (2), Weak Authentication Method (51), Content Security Policy (CSP) Header Not Set (61), Cookie No HttpOnly Flag, Cookie Without SameSite Attribute, Hash Disclosure - MD4 / MD5 (64), Hash Disclosure - SHA-1 (9), Hash Disclosure - SHA-224 (4), Hash Disclosure - SHA-256 (3), Information Disclosure - Debug Error Messages (4), Password Autocomplete in Browser, Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) (91), Server Leaks Version Information via "Server" HTTP Response Header Field (93), Web Browser XSS Protection Not Enabled, X-Content-Type-Options Header Missing (2), Base64 Disclosure (115), Cookie Slack Detector, and Information Disclosure - Sensitive Informations in URL (26).

**OWASP (ZAP)**

# Understanding PT Findings - Complex Apps -



Small apps, small problems. / Big apps, big problems!





- Too many findings... before launch date
- What that finding means, actually?
- Are there other apps affected?
- Where in the code to fix it?
- Do we have experts?



Who says it's a weakness?



### Benchmark

OWASP Top 10

OWASP Top 10 Mobile

PCI DSS

Mitre CWE

SANS Top 25

FISMA

HIPAA

MISRA

BSIMM

NIST SP 800-53

DISA STIG 4.1

WASC 2.0

OWASP ASVS

- Many security standards
- Groups of experts do a great job for us
- Mandatory or not, we have to follow them
- Not easy to know details of all standards
- Static Application Security Testing (SAST) solutions use them for code review

# Simple Q: SHA1 is OK?



February 23, 2017

Announcing the first SHA1 collision

<https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html>

“Today, more than 20 years after of SHA-1 was first introduced, we are announcing the first practical technique for generating a collision.”

## We are not all cryptologists!

**SHattered**  
The first concrete collision attack against SHA-1  
<https://shattered.io>

*A collision is when two different documents have the same hash fingerprint*

 Doc 1	 SHA-1	 42C1..21	 bad doc 1	 SHA-1	 3713..42
 Doc 2	 SHA-1	 3E2A..AE	 bad doc 2	 SHA-1	 3713..42
Normal behavior - different hashes			Collision - same hashes		

**Potentially Impacted Systems**

- Document signature
- HTTPS certificate
- Version control (git)
- Backup System



## What can SAST scanners identify?

- Vulnerabilities in the code (sample from Find Security Bugs):

```
MessageDigest sha1Digest = MessageDigest.getInstance("SHA1");
```

- How these vulnerabilities are propagated in the application

```
sha1Digest.update(password.getBytes());  
byte[] hashValue = sha1Digest.digest();
```

- Which security standards are not fulfilled

OWASP Top 10, SANS Top 25, ...

# Not Simple Q: PBEWith... is OK?



**OWASP**

The Open Web Application Security Project

- ▶ A 2: Broken Authentication and Session Management (61)
- ▶ A 3: XSS (69)
- ▶ A 4: Insecure Direct Object References (17)
- ▶ A 5: Security Misconfiguration (10)
- ▶ A 6: Sensitive Data Exposure (32)
  - ▶ Cryptographic Algorithms Used in Project (7)
  - ▶ Cryptographic Algorithms w/o Specified Crypto-Provider (7)
    - ▶ Rating: 1.00 (7)
      - EncodingLesson.java:319 - Cryptographic Algorithms w/
      - EncodingLesson.java:321 - Cryptographic Algorithms w/
      - EncodingLesson.java:364 - Cryptographic Algorithms w/
      - EncodingLesson.java:366 - Cryptographic Algorithms w/
      - EncodingLesson.java:461 - Cryptographic Algorithms w/
      - EncodingLesson.java:485 - Cryptographic Algorithms w/
      - HttpOnly.java:186 - Cryptographic Algorithms w/o Spec
  - ▶ Weak Hash Algorithms (1)

```
355
356 public static synchronized String encryptString(String str, String pw) throws SecurityException
357 {
358
359     try
360     {
361
362         PBEPParameterSpec ps = new javax.crypto.spec.PBEPParameterSpec(salt, 20);
363
364         SecretKeyFactory kf = SecretKeyFactory.getInstance("PBEWithMD5AndDES");
365
366         Cipher passwordEncryptCipher = Cipher.getInstance("PBEWithMD5AndDES/CBC/PKCS5Padding");
367
368         char[] pass = pw.toCharArray();
369
370         SecretKey k = kf.generateSecret(new javax.crypto.spec.PBEKeySpec(pass));
371
372         passwordEncryptCipher.init(Cipher.ENCRYPT_MODE, k, ps);
373
```

Simple A: PBEWith... is not OK!

```
SecretKeyFactory kf = SecretKeyFactory.getInstance("PBEWithMD5AndDES");
```

```
Cipher passwordEncryptCipher = Cipher.getInstance("PBEWithMD5AndDES/CBC/PKCS5Padding");
```

Finding Lo...

```
385     }
386
387 }
```

Solution... localized in the code.



After several minutes of research...



## Bouncy Castle

is a powerful and complete cryptography package.

```
StandardPBESStringEncryptor myFirstEncryptor = new StandardPBESStringEncryptor();  
myFirstEncryptor.setProvider(new BouncyCastleProvider());  
myFirstEncryptor.setAlgorithm("PBEWITHSHA256AND128BITAES-CBC-BC");
```

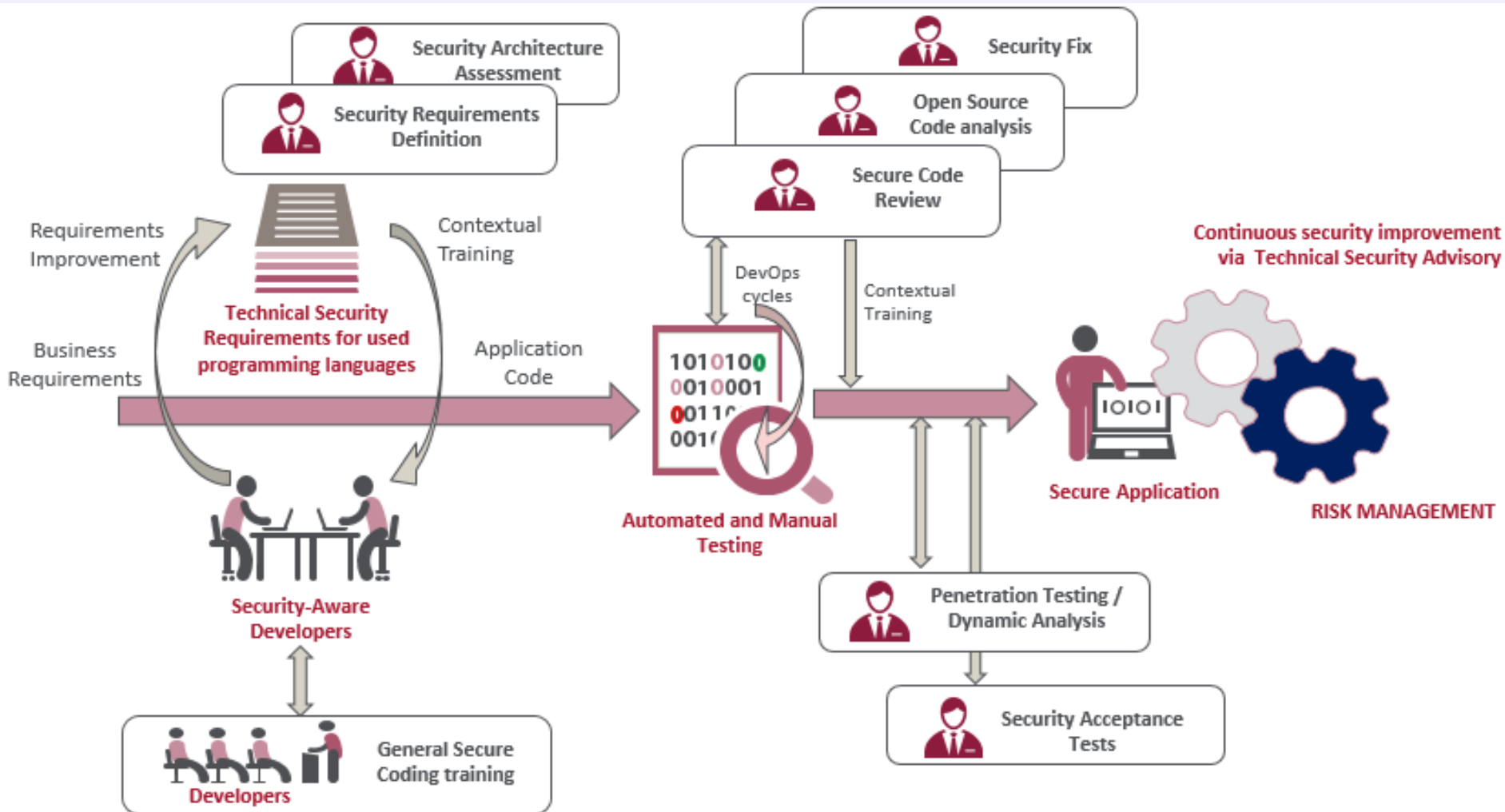


500 findings listed, 0 filtered  
Analyzed: 2017-03-31 10:35:02

Tags:OWASP Top 10	Problem Type	Rating	<>	Category	Classification	CWE Number	Reviewed State	Date
▶	A 1: Injection	(74)						
▶	A 2: Broken Authentication and Session Management	(81)						
▶	A 3: XSS	(69)						
▶	A 4: Insecure Direct Object References	(17)						
▶	A 5: Security Misconfiguration	(10)						
▶	A 6: Sensitive Data Exposure	(32)						
▶	A 9: Using Components with Known Vulnerabilities	(21)						
▶	A10: Unvalidated Redirects and Forwards	(3)						
▲	<none>	(193)						
▶	🔴 Applied Java Reflection	(4)						
▶	🔴 Usage of 'java.util.Random'	(2)						
▶	🔴 IO Stream Resource Leak	(71)						
▶	🔴 Socket Resource Leak	(2)						
▶	🔴 Trust Boundary Violation: HTTP Session	(5)						
▶	🔴 FindSecBugs: Cipher is susceptible to Padding Oracle	(2)						
▶	🔴 FindSecBugs: Cipher with no integrity	(2)						
▶	🔴 FindSecBugs: Cookie without the HttpOnly flag	(4)						
▶	🔴 FindSecBugs: Potential XPath Injection	(1)						
▶	🔴 FindSecBugs: Predictable pseudorandom number generator	(2)						
▶	🔴 FindSecBugs: Regex DOS (ReDOS)	(2)						
▶	🔴 FindSecBugs: Tainted filename read	(6)						
▶	🔴 Findbugs: Class defines equals() and uses Object.hashCode()	(2)						
▶	🔴 Findbugs: Class inherits equals() and uses Object.hashCode()	(73)						
▶	🔴 Findbugs: Field isn't final and can't be protected from malicious code	(1)						

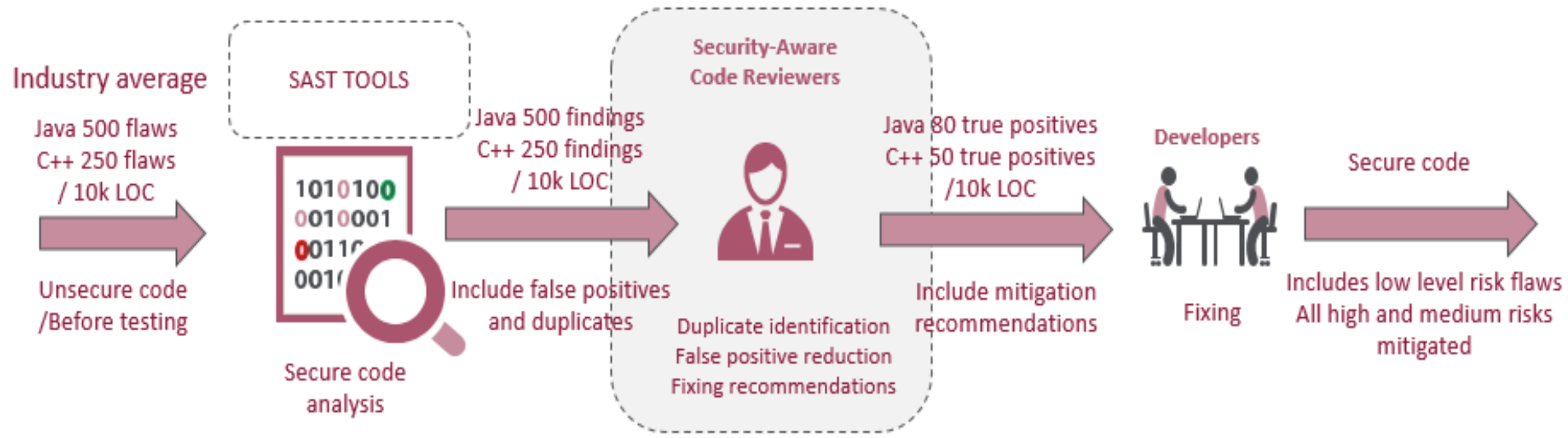
- White-box Analysis with full access to source code
- Hard to exploit everything during PT exercises
- Results can be used by PT teams for white-box testing
- Compensate lack of application security skills

# App Sec: Holistic view





# Small surface left for PT



- After fixing the code, there is a small attack surface left
- Can be input to white box / crowd security testing services
- Will challenge the skilled pen-testers

# Secure code review benefits



- Provides full context of vulnerabilities
- Compensates lack of security skills
- Is developer-friendly
- Complements PT
- Still harmless
- Timely



Thank you!

Sergiu ZAHARIA

<https://www.linkedin.com/in/sergiuzaharia/>