

Overcoming the Quality vs. Quantity Problem in Software Security Testing

OWASP AppSec 2012

Rafal M. Los (“@Wh1t3Rabbit”)
Chief Security Evangelist, HP Software



Quantity vs. Quality

...can't we just have both?

If only it was *that easy*.



Intended audience:

Corporate software security testers, professionals, or those otherwise struggling with “too many applications, not enough time”.



what's the **goal**?



why are we here?



better software



better *performance*



better ... FEATURES*



*no features, no app



better security



how to get there?



we influence security



let's start there



raise level of assurance



trust your software



validate by testing



validate by ^{security} ^ testing



millions of applications



apps in your org?



100?



1,000?



more?



no idea?



increase/decrease daily?



legacy applications



new applications



open-sourced apps



3rd party apps



composite apps



overflow by applications



they multiply!



“rapid app. delivery”



Agile, DevOps, ...



wait, wait, stop ...



who's doing the testing?



insourced, outsourced



crowd-sourced



testing window shrinkage



this causes anxiety



“security takes too long”



up against

**D
E
A
D
L
I
N
E
S**



security is 'stuck'



you must choose.



be thorough



OR



be *speedy*



“both” is very difficult



yet...



ideally you need **both**



you have 3 weapons



rely on strategy



rely on experience



rely on technology



challenge: complexity



challenge: resources



is compromise OK?



quick example



1 page, simple form



2 drop-downs



10 options each



1 check box (optional)



10 x 10 x 2 = 200 options



2 exit paths from form



Path A, Path B



198 combinations go A



2 combinations go B



Odds of testing A & B?



not very good...



it's never, *ever* this easy



tech to the rescue?



completeness vs. speed



pick most critical



experience teaches...



speed > completeness



why speed?



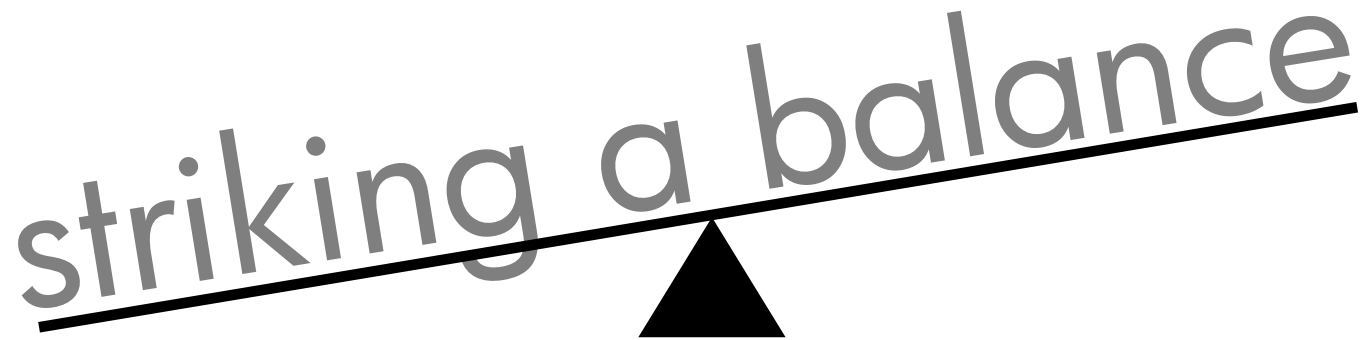
projects will go live



some testing > none



striking a balance

A graphic of a balance scale. A thick black horizontal line represents the beam, which is tilted upwards from left to right. A solid black triangle is positioned below the beam, acting as a fulcrum. The text "striking a balance" is written in a grey, sans-serif font across the beam, following its upward slope.

balance is difficult



experience has taught me



balance **is** possible



what matters?



Size of application



application **C**omplexity



available **T**ime to test



available **R**esources



other considerations



release cycle



application criticality



external support



strategies for balance



1. app segmentation



split the application up



at functional|boundaries



apps break into pieces



e.g. register → login



e.g. login → catalog



at trust|boundaries



high | $\triangleleft \triangleright$ | low security



unauth | auth



along workflows



use case :: workflow



psst! don't forget data



2. workload distribution



technology



win



evolving technology



harness compute power



compute in the cloud



vLIMITLESS resources



massively parallel



instant-on scan farms



start on your desktop



push to the cloud



pay for compute time



time: days → hours



billions of iterations



Yay cloud!



3. OPTIMIZE tools



tools give us options



use advanced options





JavaScript operations



number of iterative loops



form variations/submits



page timeouts.....

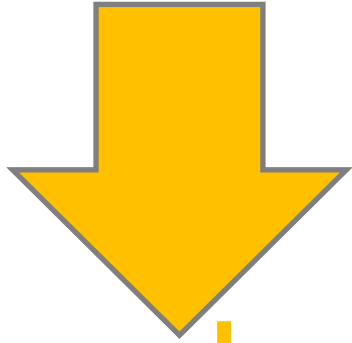


redundant pages?
redundant
redundant
redundant



cASe SenSITivity





“start here” capability



many other tech controls



to recap



applications → complex



speed vs. completeness



looking for balance



AS appropriate



balance **is** possible



have cake, eat it too.



3 ways to achieve



app segmentation



workload distribution



tools optimization



ANY
QUESTIONS?



THANK YOU

