# Testing Large Number of Applications

Notes from the Field

## OWASP AppSec India 2008 Conference

New Delhi – Aug 2008


**Roshen Chandran**

Director, Paladion

roshen.chandran@paladion.net

+91 93416 36152



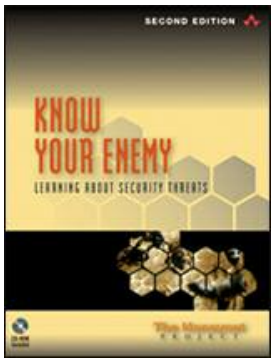


**The OWASP Foundation**

http://www.owasp.org/

# Speaker Intro

- Roshen Chandran
  - Director - Application Security Practice, Paladion
  - Co-author "Know Your Enemy" from Addison Wesley
  - Experience in 50+ penetration tests and code reviews
  - Contributor to the OWASP Appsec FAQ
  - Presented at RSA, CansecWest, SDWest



Trend #1

**Testing large number of applications in the enterprise**

- Our largest testing project
  - 20 Apps in 2003
  - 500 Apps in 2007
- More enterprises want to test 50 – 250 apps



# What's driving the trend?

- Primarily, compliance requirements
  - Especially SOX
  - Both in the US and in India
- Fed by greater awareness
  - Thanks to forums like OWASP



# Where is this seen most today?

- > $ 500 million enterprises
- Global footprint
- 3 we've seen with large number of apps are from Banking and Finance
  - The case study today is generic and sanitized



# The types of apps

- Mix of internal and external facing apps
  - 90 - 10
- Mix of web apps and thick clients
  - 70 - 30



Trend #2

**From business owners to centralized teams**

- The control is moving to centralized teams
- The benefits
  - Standardize the approach
  - Co-ordinate the tests better
  - Negotiate better rates



Ground reality #1

**Budgets are limited**

- Application security testing budget is a small slice of the overall security budget



Ground reality #2

**Internal expertise might be limited**

- While the Security officers were fluent in
  - Application security best practices
  - OWASP Top 10
- Beyond them, the awareness was a lot lesser
- Testing expertise is usually in short supply



## But the good news…

- Strong program management capabilities
- Visible support from Top Management
- Visionary leadership



# Traditional penetration tests don't scale

- Traditional pen tests take 10 – 15 days  per app
  - Is the gold standard of security testing today
  - Is expensive



# The 4-step professional pen tests

1. Create Threat Profile
2. Create the Test Plan
3. Execute the Tests
4. Prepare the report



# The Threat Profile

- Threat = "goal of the adversary"
- Threat Profile is the list of all threats to app
- Example – for a travel booking site
  - An adversary…
    - Tricks others to buy tickets at a higher price
    - Buys tickets at a lower rate than advertised
    - Modifies the itinerary of another user
    - Cancels the tickets of other users
    - Views the itinerary of other users
    - Shuts down the site
    - …



# The Test Plan

- Maps each threat to relevant pages and the relevant attacks
- Example
  - Tricks others to buy tickets at a higher price
    - Relevant page(s): Buy ticket, Confirm purchase
    - Relevant attacks: Variable manipulation, CSRF
- Converts the Threat Profile into meaningful attacks



# Test Execution

- Combination of manual and automated techniques
- Automated testing
  - Injection attacks
  - Cross Site Scripting
- Manual testing
  - Business logic flaws
  - Privilege escalation



# Reporting

- Detailed reports with
  - Walk through of attack with screen shots
  - How to solve it
- A good app pen test takes time



Ground reality #3

**For 200 apps, penetration tests are too expensive**

- No one had the budgets for 200 app pen tests
  - 200 x 15 = 3000 days
- A more pragmatic approach was required



# Nor was Automated scanning enough

- Automated scanning is quick, inexpensive
- It's getting better over the years
- In isolation, not useful
- Enterprises want more than a scanner output



# How they attacked the problem

1. Different levels of testing
2. Framework for classifying apps
3. Baseline standard checklist
4. Streamline reporting



# Different levels of testing

- All apps won't undergo the same level of testing
  - Some apps will get a full test
  - Others will get a shorter, faster test
- Purists wouldn't like it, but this was pragmatic
  - With limited budgets, how do we test 200 apps best?



# Framework to classify apps

- A risk assessment framework to prioritize apps
  - Some were simple, some were complex
  - Ultimately, some simple questions
    - Does the app face the internet?
    - Does the app handle customer sensitive data?
    - What's the potential financial loss if the app is down for a day?
- Prioritizing helps share the limited budget better between the apps



# Baseline standard for the security tests

- A minimum set of checks for all apps
  - Does it do input validations at the server?
  - Does the app adhere to the password policy?
  - Does it check for old password when changing the password?
  - Is it safe against SQL Inj, XSS, CSRF?
- Typically 40 – 70 checks
- Not much of privilege escalation attacks
- Threat profile not essential for these tests



# Streamlined Reporting

- Simplified the template
  - Summary - Simple spreadsheet
  - Detailed – No screen shots
- Report Repository
  - Readymade description/solution of common findings
  - Easy to copy-paste from
  - Minimal tailoring required for each finding
- Reporting time dropped to 2 hours



# Initial estimates

- Duration
  - Baseline Security Test: 2 days
  - Detailed Security Test: 4 – 8 days

- % of Apps
  - Baseline Security Test: 50%
  - Detailed Security Test: 50%



## For 200 apps, the total would be

100 x 2
+ 100 x 5
= 700 person days

- That would fit the budget, if that worked according to the plan



# Lessons from the 1st month pilot

- What worked
  - The streamlined reporting worked
  - The baseline tests ~ 1 day
- What didn't
  - The 5 day detailed test was a stretch
- And we still missed the target by 20%



# The vital lesson from the pilot

- Starting delays are a culprit
  - Hidden
  - Pernicious
- For a 2-day test, a 1-day delay is a 50% hit on schedule



# Countering "start delays"

- Strong program management
  - Schedule 4 weeks in advance
  - Follow up frequently in that time
  - Over-book by 20%
    - As some logins might not come anyway



# New challenges as the numbers picked up

1. Profusion of reports
2. A few laid-back app owners
3. Peaks and troughs in load



# Profusion of reports

- Too many password protected PDFs floating around
- The Solution
  - Online reporting portal with logins to business owners
  - Reports could be exported to PDF when required



# A few laid-back app owners

- Some app owners couldn't make the time
  - They already had a lot on their hands
- Solution
  - Escalate during reviews with senior management
  - Visible support from senior management
  - Those lagging were pulled up



# Peaks and troughs in load

- Schedules are made 4 weeks in advance
  - But, apps might miss the schedule
- About 2 weeks visibility into future load
  - Team to resize dynamically with 2 weeks notice
- Not very easy
- And seating space ☺



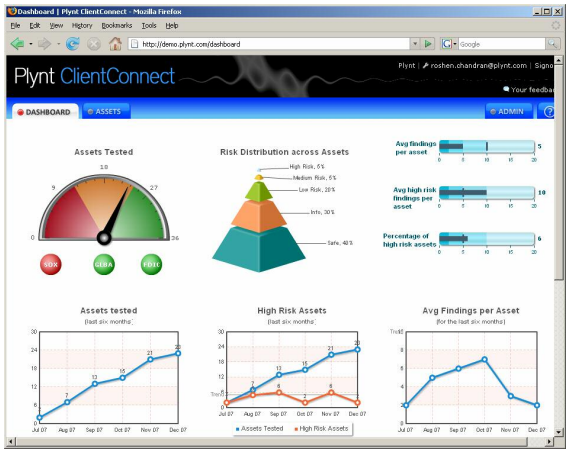
# Crossed the 100th app in 4th month

- Quite thrilled
  - This was a comforting milestone
- This was 3 weeks behind initial schedule
- The half-way mark changes the outlook



# Metrics program

- Quantitative feedback on the benefits
- Ideal if started from the first
- In practice, it started after the half way mark
- Common metrics
  - Average no. of vulns per app
  - No. of vulns closed per month
  - Distribution of risk profile of vulns
- Touched 200th app in 8th month






# Summary

1. Classify apps based on risk
2. Different levels of testing
3. Standardize on the baseline test
4. Reduce waste – streamline reporting
5. Schedule in advance
6. Consider online reporting
7. Get reviewed by senior management
8. Work closely with partners to manage fluctuating loads
9. Metrics program to measure effectiveness



# Thank You