Paco Hope <paco@cigital.com>
Florence Mottay <fmottay@cigital.com>

SECURING THIRD PARTY SOFTWARE

Objectives

- Define third party software
  - What it is, why we use it
- Define the risks from third party software
- Identify tools and techniques for addressing the risk
- Connect tools with the right situations
- Explore one possible approach

Software Ecosystem

Also Known As: Software Supply Chain Risk

- Studied by Software Engineering Institute (SEI) and University of Maryland
- Who supplies code?
- Who supplies labour?
- Who operates software?

Kinds of "Third Parties"

1. Staff augmentation — they work on your premises, they follow your orders
2. Integrated Project Teams — they provide a team; you provide management
3. Contractual — You write contracts; they deliver
4. Service Provider — You buy what they sell

The Problem

You can outsource activities, but you cannot outsource liability

- Risk introduced by software
  - How do we identify it?
  - How do we quantify/qualify it?
- Various activities address that risk
  - How do we conduct those activities in someone else's lifecycle?
  - How do we hold the right people accountable?

You acknowledge that Licensed Software is not designed or intended for use in the design, construction, operation or maintenance of any nuclear facility. Sun Microsystems, Inc. disclaims any express or implied warranty of fitness for such uses.

Risks from Software

- Brand and reputation damage
- Non-compliance
- Failures in business logic
  - Lost sales
  - Unauthorised disclosure of data
  - Unavailability
- Actual back doors or vulnerabilities

IANAL, But... (I am not a lawyer)

Popular Open Source Licenses

- Apache
- "Artistic" License
- BSD
- GNU General Public License
- GNU "Lesser" GPL

Possible Implications

- Releasing source code
- Permitting derivative works
- Disclosing origin of software

http://www.opensource.org/licenses

Thinking About the Actors

Them

- Do our vendors have a clue?
  - When designing
  - When coding
  - When operating
- Do they do their jobs well?
- Are their products suitable for us?

Us

- Do we have a clue?
  - Requirements
  - Operations
  - Support
- Are we doing our part well?
  - Integration
  - Compliance

Tools for the Problem

- Assessing capabilities
- Deliverable-based security gates
  - Security requirements
  - Security test plan
  - Threat model
  - Code Scan results w/ defect tracking
  - Security test results mapping to requirements
  - Penetration test results
- Contract-based hooks

Build Security In Maturity Model – BSIMM

For Us

- What do we do?
- How mature are we?
- Where might we put more effort?

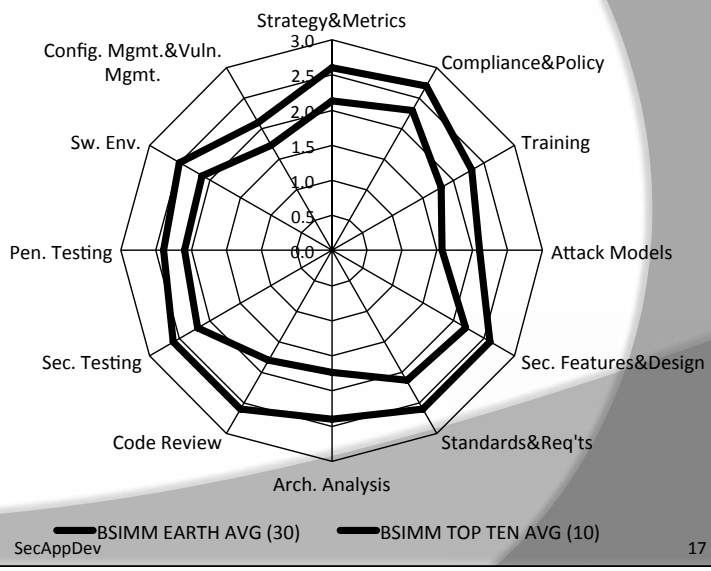
For Vendors: vBSIMM

- Quick / crude measure
- 15 activities
- Very low bar
- Vendors still score poorly

http://www.bsimm.com/

Learning What Others Do

- identify gates
- unify regulations
- know PII obligations
- publish policy
- awareness training
- data classification
- identify features
- security standards
- review security features
- static analysis tool
- QA boundary testing
- external pen testers
- good network security
- incident response
- close ops bugs loop



* ("everybody" = 20 out of 30 firms)

Alternatives to Consider

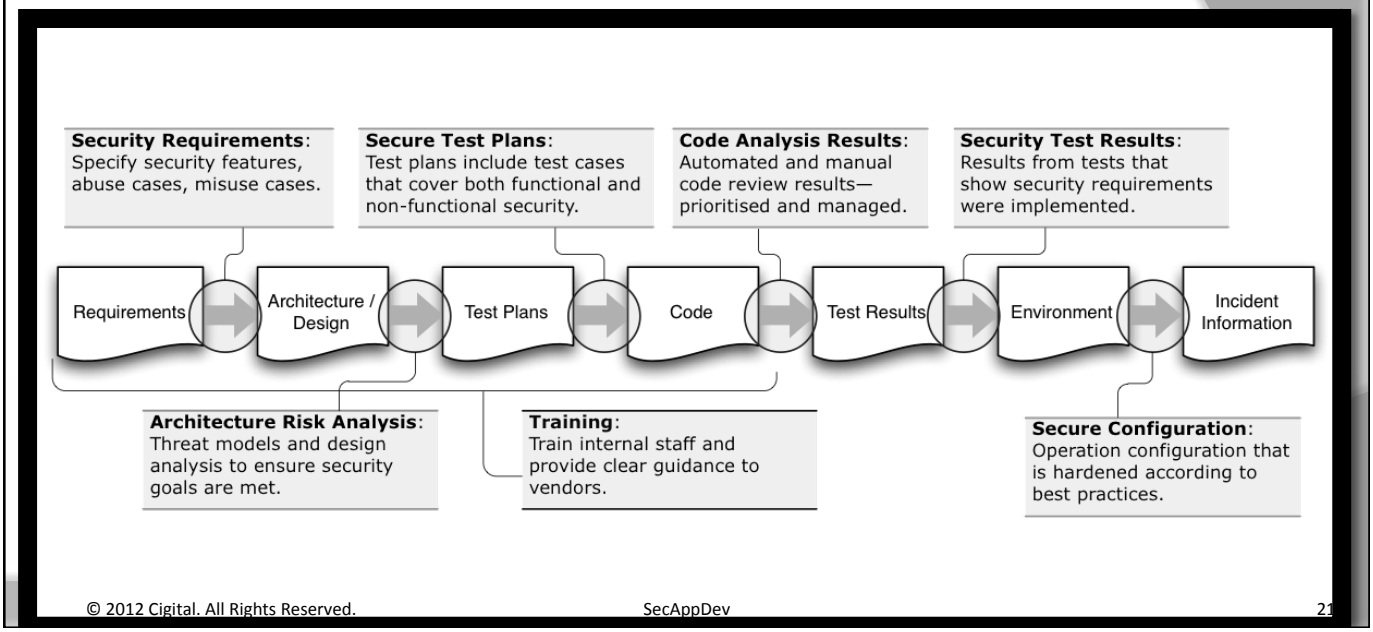
- Microsoft SDL
- OpenSAMM
- CLASP
- Etc.

- Tend to be "prescriptive" not "descriptive"
- Don't help you measure yourself or others

Tools and Situations

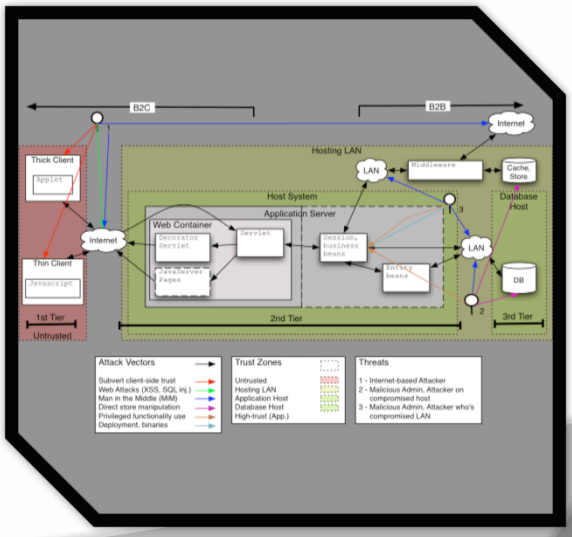
- We can often apply security requirements
  - Very applicable when we specify
  - Harder to enforce in SaaS—limited by vendor's flexibility
- Code scanning is very good evidence
  - Only works when you have code
  - Binary scanning is a poor substitute
- Security testing always possible in UAT
- Pen testing requires cooperation, often limited scope

Touchpoints for Third Party Development



Architecture Deliverables

- Architecture risk analysis
- Threat model
- Test strategy and test plan with security



Use the Source, Luke

- Static analysis
- Defect tracking
- Patch management

- OSS analysis
  - Identify accidental / unknown usage
  - Identify legal obligations

- Static Analysis tools
  - Commercial
    - Fortify
    - Coverity
    - AppScan Source
  - Free
    - CppCheck
    - Findbugs
- OSS Analysis
  - Black Duck
  - Palamida

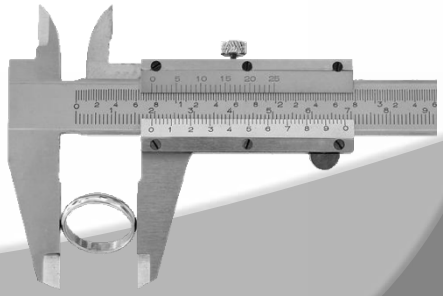
Working with Binaries

- Reverse engineering
  - Good for mobile, embedded, client/ server
  - Not always permitted
- Binary analysis
  - Veracode, etc.
  - Not always successful

- Simulation
  - Run in VM, sandbox, or simulator
- Observation
  - Eavesdropping
  - Proxying,etc.

Security Testing (Not Penetration Testing!)

- Boundary cases
- Negative cases
- Inverted cases off the RBAC matrix
- Identifying undesirable behaviour
- Not "checking"
- Not functional testing
- Exploratory testing is good



Penetration Testing

- Require vendor support
  - access credentials
  - generous time windows
  - etc.
- Require vendor tracking / reporting
  - What will they do?
  - When will they do it?
- Focus on solutions, not problems
  - Not about finding max bugs
  - It's about fixing bugs

Deployment

- Secure configuration
- Change control process
- Coordination with development team
- Upgrades to base platform
- Patch deployment to application

Operations

- Logging, monitoring SIEM
- Incident response
- Vulnerability tracking

Procurement

- Security requirements during RFP / Tender process
- Security questions during vendor selection
- Periodic evaluation of vendor security capabilities
- Security deliverables with functional deliverables

Procurement

- Require source code
- Permit decompiling / reverse engineering
- Permit security testing
- Require significant documentation
- Escrow code if you must

One possible flow

- Figure out which lifecycle stages are out of your control
- Figure out which deliverables are feasible
- Identify mechanisms to enforce deliverables (e.g., UAT, procurement, etc)
- Require deliverables at appropriate stages
- Add to PMO process, if possible

1: Identify Lifecycle Stages

- Which ones are owned by vendors?
- Where do your teams plug in?
  - PMO
  - Project leads
  - Procurement
  - Requirements
  - Integration
  - UAT

2: Identify Practical Deliverables

What?

- What is practical, permissible, measurable?
  - Threat models
  - Code scan reports
  - Pen test reports
  - Defect reports
- Is it objective?
- How much visibility do you get into its creation?

When?

- At major releases?
- At regular intervals?
- On-demand access?
- As it is generated, or after it is reviewed by the vendor?

3: Promote Enforcement

Lifecycle Phases — Promotion

- Create security gates
  - Dev → QA
  - QA → Staging
  - Staging → Production
- Require security deliverables for promotion phase-to-phase
- Enforce security sign-off

Be Pragmatic

- If "security" always says 'no', then "security" becomes a *problem*
- Problems get "fixed"
- Choose battles carefully

4: Institutionalise

Put Security in the PMO

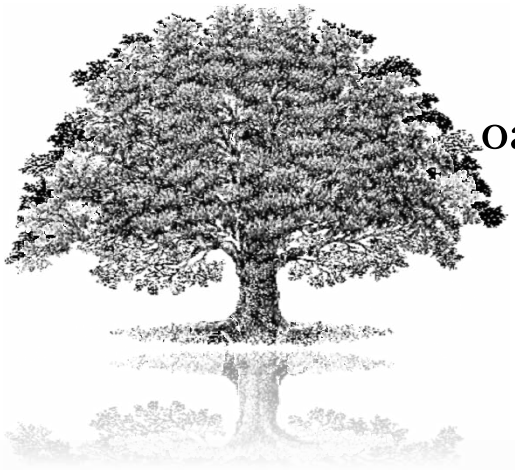
- Make it regular
- Make it understood
- Automate as much as possible
  - Checklists
  - Worksheets
  - Processes

Change Takes Time

- Start small
- Minimise overhead
- Make everything relevant
- Ensure adoption of one small piece before introducing a new piece

Putting it Together

- Identify your vendors and sources of third party software risk
- Understand your competency and theirs
- Determine ownership of lifecycle phases
- Identify security deliverables for each phase
- Gradually work them into your process



The best time to plant an oak tree was twenty years ago.

The next best time is now.

—Ancient Proverb

Paco Hope <paco@cigital.com>
Florence Mottay <fmottay@cigital.com>