RSA Conference2016
San Francisco | February 29 – March 4 | Moscone Center

Connect to Protect

SESSION ID: STR-W05

Estimating Development Security Maturity in About an Hour

Matt Clapham
Principal, Product Security
GE Healthcare
@ProdSec

How We Got to Now

- Needed something given volume
- 4 things recommended
- Added one more
- Assessment leverages experience
- Field-tested with suppliers and devs

Not for n00bs

- Security operations
- Development experience
- Testing (penetration, vulnerabilities, security, or in general)
- Cryptography background
- Incident response
- Certifications
- Etc.

Three Example Stories

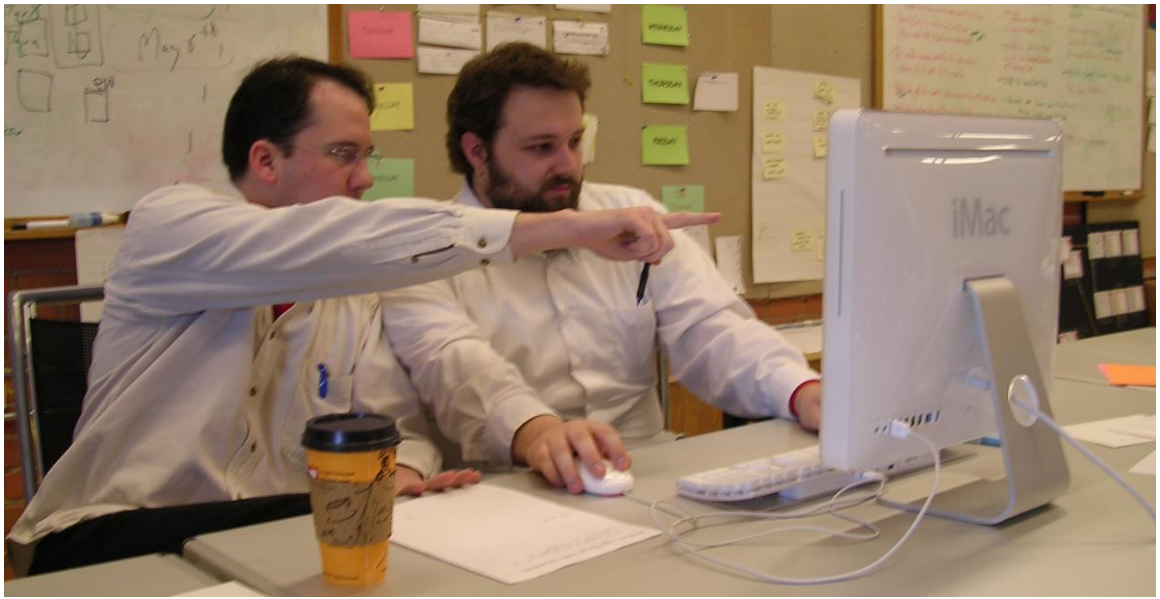
- The 11th-hour Security Review
- The Acquisition
- The Internal Development Team

The 11th Hour Security Review

"This security review won't take long, will it?"

The Acquisition

"So we're going to buying company *X* for its ___ product."



The Internal Development Team

"Those folks making that software over there…"

Agenda

- Overview
- Five key behaviors
- Scoring and report
- Pros, cons, and other models
- Extending the method
- Summary
- Application

Method Overview

- Pre-meeting research - 5 to 15 minutes
- Meet with development team - 30 to 60 minutes
- Score and report back - 15 minutes

Goal: 90+% confidence on assessment

Total time: 50 to 90 minutes or “About an hour on average.”

Pre-meeting Research

- Process time required: 5 to 15 minutes
- Looking for…
  - Public details about vulnerabilities
  - Company responses and published guidance
  - History of supporting and fixing the software
- Search on software name or company name and…
  - NVD
  - vulnerabilities
  - patches

Meet with the Development Team

- Process time required: 30 to 60 minutes
- Resources needed: developers, testers, and program managers
- Goals
  - Establish rapport
  - Status on each of the 5 behaviors
  - Additional information on how software is made securely

Development Meeting Sample Agenda

- Introductions and establish rapport – 5 minutes
- Identify necessary resources – 1 minute
- Technology overview – 3 minutes
- Probing questions on each focus area – 20+ minutes
- Wrap-up – 2 minutes

Development Meeting Warning Signs

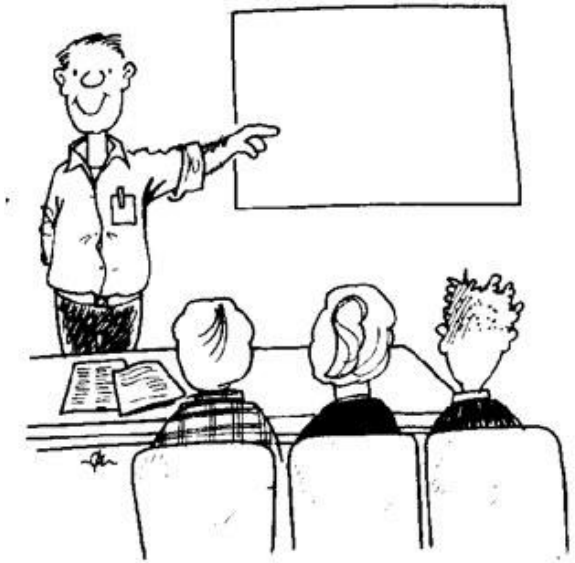
- “I’m just the IT guy.” or “I’m the CISO.”
- “We can’t tell you because that would be a security risk.”
- “Our software has no flaws.”
- “We wrote our own encryption method…”
- “Bob does that for us and he’s on vacation.”

Five Key Behaviors

- Developer training in secure development
- Design threat modeling for security refinement
- Static code analysis
- Dynamic analysis and testing
- Vulnerability and incident response process

What to Look For in Each

- 90% confident on a yes/no answer
- Anecdotes regarding technology, design pattern, etc.
- Artifacts (extra points if publicly available)
- Positives, i.e. increased quality or above and beyond
- Negatives, i.e. poor execution or follow-through
- Nearest exit, i.e. a statement that indicates yes/no quickly



Behavior 1: Secure Development Training

Do you train developers in writing secure code?

Training: Openers

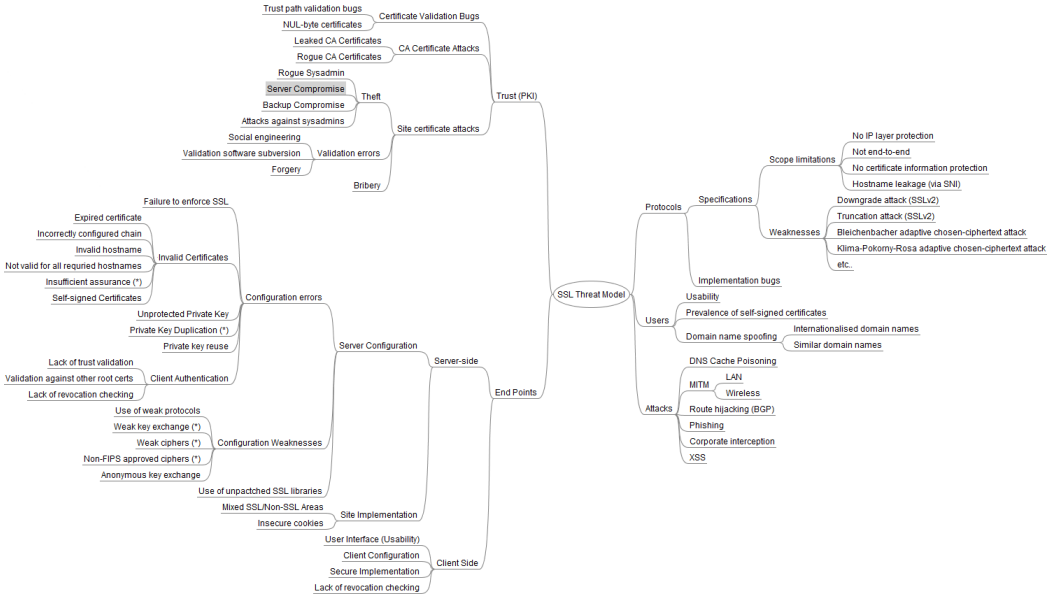
- How do you teach a new developer to make a secure product?
- How would your developer know which security tech to use?
- Describe the security training developers receive.
- How do you keep developers up to date on security risks?

Training: Positives and Negatives

- Positives
  - Regular security conference attendance
  - Internal or External professional training
- Negatives
  - Focus on general secure computer usage awareness
  - Single annual conference attendance

Training: Warning Signs and Exits

- “Every developer has a chat with our Sr. architect.”
- “Our compliance team requires…”
- “We don’t need to worry about that because…”
- “Our cloud provider takes care of defending us.”
- “We sent Bob to a conference last year and he gave us slides.”



Behavior 2: Threat Modeling

How do you refine the design to remove threats?

Threat Modeling: Openers

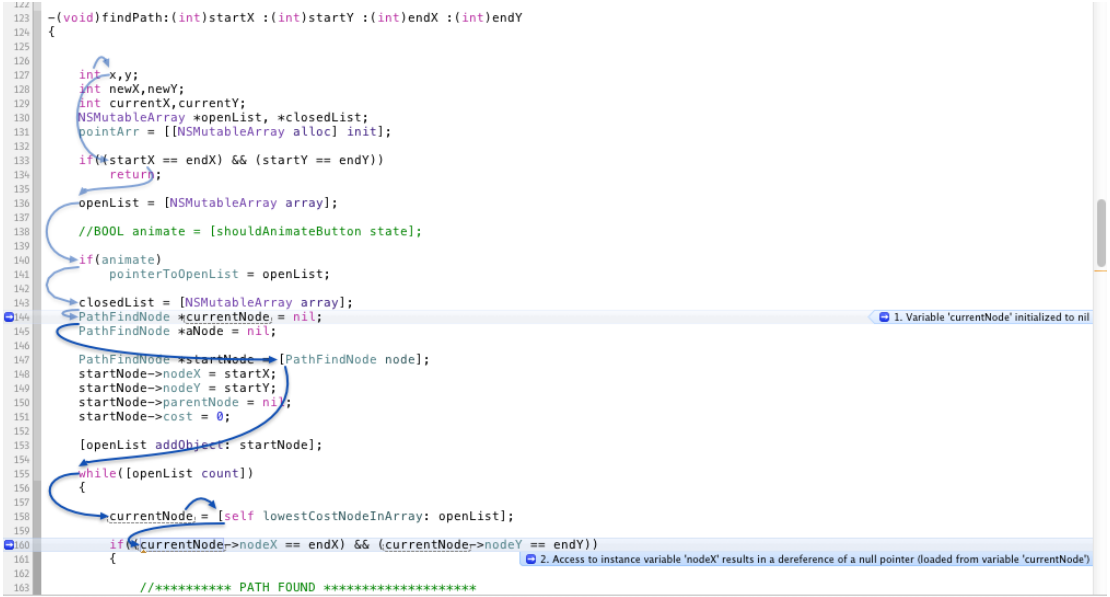
- Describe your group's threat modeling process.
- What tools do you use to model threats to the application?
- How do you refine the security of the design?
- How do you know when you're done threat modeling?

Threat Modeling: Positives and Negatives

- Positives
  - Formal process
  - Uses tool(s)
  - Always up to date
- Negatives
  - Ad hoc
  - Dependent on one person

Threat Modeling: Warning Signs and Exits

- “A couple of us get together and review the design.”
- “Bob does that for us and he’s on vacation.”
- “We had a penetration test last year and…”
- “We use SSL.”




```
-(void)findPath:(int)startX :(int)startY :(int)endX :(int)endY
{


    int x,y;
    int newX,newY;
    int currentX,currentY;
    NSMutableArray *openList, *closedList;
    pointArr = [[NSMutableArray alloc] init];

    if((startX == endX) && (startY == endY))
        return;

    openList = [NSMutableArray array];

    //BOOL animate = [shouldAnimateButton state];

    if(animate)
        pointerToOpenList = openList;

    closedList = [NSMutableArray array];
    PathFindNode *currentNode = nil;
    PathFindNode *aNode = nil;

    PathFindNode *startNode = [PathFindNode node];
    startNode->nodeX = startX;
    startNode->nodeY = startY;
    startNode->parentNode = nil;
    startNode->cost = 0;

    [openList addObject: startNode];

    while([openList count])
    {

        currentNode = [self lowestCostNodeInArray: openList];

        if((currentNode->nodeX == endX) && (currentNode->nodeY == endY))
        {

            //********** PATH FOUND ********************
```

Behavior 3: Static Code Analysis

How do you review your code for security flaws?

Static Analysis: Openers

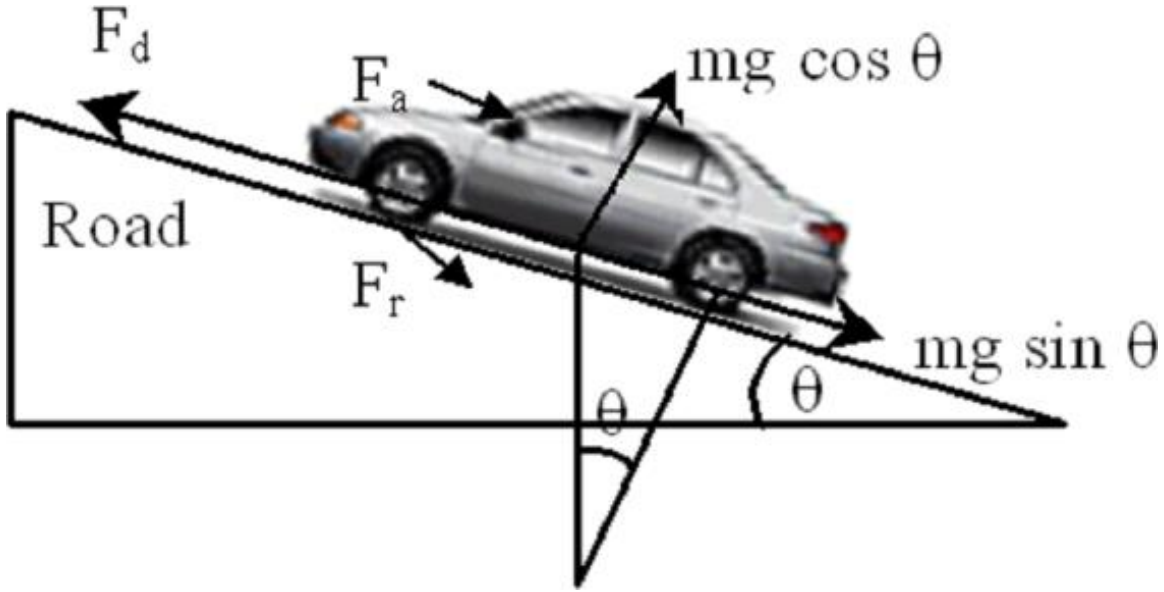
- What code analysis tools do you use?
- How do you check your code for security flaws?
- How would you know if there's a buffer overflow in your code?
- Describe your code review process.
- How do you know your code is secure?

Static: Positives & Negatives

- Positives
  - Build integrated
  - Developer client
  - Clean check-in requirement
- Negatives
  - Manual or ad hoc execution
  - No fixing of identified problems
  - No security tests in testing suite

Static: Warning Signs and Exits

- “We review every line of code manually.”
- “Bob reviews all our code himself.”
- “We use lint.”
- “We compile with all warnings on.”



Behavior 4: Dynamic Analysis

How do you test your software for security flaws?

Dynamic Analysis: Openers

- How do you know if your web application has a cross-site scripting vulnerability or not?
- How do you find vulnerabilities that have slipped through the cracks?
- Tell me about how you test the security of your service.
- Do you use any tools to automatically test the security of your application?

Dynamic: Positives and Negatives

- Positives
  - Regularly scans production
  - Coordinated with Static analysis
  - Quality gating function
- Negatives
  - Only uses a configuration scanner
  - Only focuses on OWASP top 10

Dynamic: Warning Signs and Exits

- “Once a year we hire a security firm to penetration test.”
- “Bob scans our systems for missing patches.”
- “We use a patch scanner to test for vulnerabilities.”
- “We tried using a tool but it broke our software.”

Behavior 5: Vulnerability and Incident Response (V&IR)

Who gets the call at 3AM if there's a vulnerability to fix?

V&IR: Openers

- How would a researcher inform you of a problem and confirm that you've fixed a vulnerability?
- How can the operations team tell if a vulnerability is in the code you wrote or not?
- Describe your incident response process.

V&IR: Positives and Negatives

- Positives
  - Public “trust center” for customer communications and bulletins
  - Security community engagement like bug bounties
  - Planned (and practiced) process for handling vulnerabilities
- Negatives
  - Informal or ad-hoc
  - Thinking its only about middleware or OS patches

V&IR: Warning Signs and Exits

- “Our operations are experienced in dealing with patches.”
- “We don’t scan in our production environment.”
- “No one has reported any vulnerabilities to Bob.”
- “We’re too small to be attacked.”
- “We haven’t been hacked yet.”

Scoring Method

- Review notes and evidence
- Starts at 0
- 1 point for every confirmed behavior
- Average out modifiers for Positives and Negatives
- Aggregate score is number with + / -
- Range of 0- to 5+

Simple Scoring Examples

Example 1

Aspect	Notes
Search results	Nothing
Training	Yes
Threat modeling	No, -
Static analysis	No, +
Dynamic analysis	Yes, + +
Vulnerabilities & incident response	No, -

Example 2

Aspect	Notes
Search results	Security portal
Training	Yes, + -
Threat modeling	No, -
Static analysis	Yes, + +
Dynamic analysis	Yes, + - -
Vulnerabilities & Incident Response	Yes, -

Scoring Example 1: 11th Hour Review

Aspect	Score	Notes
Search results		Minimal public presence, no detail of security at all
Training	No	Alice gave security presentation and relayed news items
Threat modeling	No	Ad hoc design review not focused on security
Static analysis	Yes	Built-in to IDE, compiler warnings as errors
Dynamic analysis	No	Alice did some ad hoc security testing using free tools
Vulnerabilities & incident response	No	Customer reported problems goto account manager first

Scoring Example 2: The Acquisition

Aspect	Score	Notes
Search results		Security info center
Training	Yes	N00bs goto professional class; industry hires optionally
Threat modeling	Yes	No tool; security architecture review team
Static analysis	No	Custom rules for style only, lacking a tool for Ruby
Dynamic analysis	Yes	Promotion requirement, bug bounty program
Vulnerabilities & incident response	Yes	Incident response for patches not home-grown code, escalation process to development team in place

Scoring Example 3: Internal Developers

Aspect	Score	Notes
Search results		Team has design docs for security features available
Training	No	Optional for testers
Threat modeling	Yes	Formal process, not all problems addressed
Static analysis	Yes	Rules don't specifically target security
Dynamic analysis	No	Tried once and it broke things
Vulnerabilities & incident response	Yes	Single point of contact is Bob, just Bob

Report Back

- Brief recap of 5 behaviors
- Refer to evidence
- Score with Positives or Negatives
- Explain the score and answer questions
- Optional feedback to the development teams

Tips & Tricks

- Be respectful and flexible in questioning
- Leverage experience to spot misdirection
- Avoid black holes in discussion
- Repeat back key responses
- Keep good notes, especially scoring justifications
- Seed memes of good behaviors

Is and Isn't

What it is…

- Speedy
- Rough
- Smoke detector
- Creator of more action items
- Anecdote for a recommendation

…and isn't

- Full maturity model
- Secure development process
- Standards based
- Compliance audit
- Complex

Building Security In Maturity Model

- a.k.a. BSIMM
- Based on actual usage (6 doz. Companies)
- Biases to assessed population
- Many behaviors considered across four domains
- Maturity levels for each item
- Takes about 2 to 4 weeks for a full evaluation

Open Software Assurance Maturity Model

- a.k.a. OpenSAMM
- Creative-Commons Licensed
- Simpler than BSIMM
- Handy score cards and templates
- Self-achievable
- Takes 2 days to 1 week

For Just a few Hours More

- Suggest trainings and resources
- Provide formal list of recommendations
- Review vulnerability history for trends
- Review the vulnerability history of the software's dependencies
- Kickoff a maturity improvement program
- Referral to formal maturity assessment

Summary

- How the method came about
- Overview
- Five key behaviors
- Scoring and reporting
- Pros, cons, and other models
- Adding a few extras

Apply What You Have Learned Today

- 3 days: Use it on a last-minute security review
- 3 weeks: Start using it to triage application security reviews
- 3 months: Formalize estimation as a process to gate further security analysis