

Vulnerability Assessment: The Assessors Experience

Barton P. Miller
James A. Kupsch

Computer Sciences Department
University of Wisconsin

Elisa Heymann

Computer Architecture & Operating
Systems Department
Universitat Autònoma de Barcelona

OGF27, Banff, Canada

October 12, 2009



1



Session Objectives

What to expect:

- Getting started - there are many reasons to say "no".
- The vulnerability assessment process - what makes our life easy or difficult.
- When the first vulnerability reports come in - what do you do?

Remember that we're on *your* side.



2



Just say “no”.

(Nancy Reagan)

There are *Lots* of Reasons to Say No

“We use best practices in secure software design, so such an effort is redundant.”

There's many a slip 'twixt cup and lip...

(old English proverb based on Erasmus)



Even the best programmer makes mistakes.

- The interaction between perfect components often can be imperfect: *falling between the cracks*.
- Even in the best of cases, only works with formal specification and verification.

There are *Lots* of Reasons to Say No

“It’s too expensive.”

The best defense is a good offense.

(old sports adage)

The only real defense is active defense.

(Mao)

- Yes, is it expensive.
- And, yes, if you are successful, you will only see an expense.
- However the cost to recover after a serious exploit is *prohibitive*.



5



There are *Lots* of Reasons to Say No

“I’ll just run some automatic tools.”

The era of procrastination, of half-measures, of soothing and baffling expedients, of delays is coming to its close. In its place we are entering a period of consequences.

(Winston Churchill, August 1941)



- Tools like Fortify and Coverity are worthwhile to use...
- ...however, don’t let them give you a false sense of security. Our recent study demonstrates their significant weaknesses:

J.A. Kupsch and B.P. Miller, “[Manual vs. Automated Vulnerability Assessment: A Case Study](#)”, *First International Workshop on Managing Insider Security Threats*, West Lafayette, IN, June 2009.



6



There are *Lots* of Reasons to Say No

“If we report bugs in our software, we will look incompetent.”

A life spent making mistakes is not only more honorable, but more useful than a life spent doing nothing.

George Bernard Shaw (1856 - 1950)



- All software has bugs.
- If a project isn't report the bugs, either they are not checking or not telling.
- Our experience shows that users (and funding agencies) are more confident when you are checking and report.

And the assessment team arrives...



During the Assessment

What makes our job harder:

- Incomplete or out-of-date documentation.
- Complex installation procedures, especially ones that are not portable and require weird configuration file magic.
- Lack of access to full source code.
- Lack of access to development team.

During the Assessment

What you can expect from us:

- We work *independently*: crucial for an unbiased assessment.
- We will ask you lots of question.
- We won't report any vulnerabilities until we're done...
 - ...however we *will* release our intermediate products - diagrams from the architectural, resource, and privilege analyses.
- It will take longer than you think...
 - ... we don't report a vulnerability until we can construct an exploit.

And then the vulnerabilities arrive...



We do Find Vulnerabilities

System	Origin	Language(s)	Size (loc)	Vuln. Found
Condor	Wisconsin	C++	600K	15
SRB	SDSC	C, SQL	275K	6
MyProxy	NCSA	C	25K	5
gLExec	Nikhef	C	43K	5

How do You Respond?

*When In Danger, When In Doubt, Run In Circles,
Scream And Shout*



How do You Respond? (really)

- Denial: “That’s just not possible in our code!”
- Anger: “Why didn’t you tell me it could be so bad?!”
- Bargaining: “We don’t have to tell anyone, do we?”
- Depression: “We’re screwed. No one will use our software and our funding agencies will cut us off.”
- Acceptance: “Let’s figure out how to fix this.”

How do You Respond?

- Identify a team member to handle vulnerability reports.
- Develop a remediation strategy:
 - Study the vulnerability report.
 - Use your knowledge of the system to try to identify other places in the code where this might exist.
 - Study the suggested remediation and formulate your response.
 - Get feedback from the assessment team on your fix - very important for the first few vulnerabilities.
- Develop a security patch release mechanism.
 - This mechanism must be separate from your release feature/upgrade releases.
 - You may have to target patches for more than one version.

How do You Respond?

Develop a notification strategy:

- What will you tell and when?
- Users are nervous during the first reports, but then become your biggest fans.
- Often a staged process:
 1. Announce the vulnerability, without details at the time you release the patch.
 2. Release full details after the user community has had a chance to update, perhaps 6-12 months later.
- Open source makes this more complicated!

The first release of the a patch reveals the details of the vulnerability.

How do You Respond?

A change of culture within the development team:

- When security becomes a first-class task, and when reports start arriving, *awareness* is significantly increased.
- This effects the way developers look at code *and* the way that they write code.
- A major landmark: when your developers start reporting vulnerabilities that they've found on their own.
- Open source makes this more complicated!

The first release of the a patch reveals the details of the vulnerability.



17



Discussion

<http://www.cs.wisc.edu/mist>

bart@cs.wisc.edu



18

