Does Software Quality Matter?

Sandy Clark, Matt Blaze, Jonathan Smith
University of Pennsylvania

Usenix Security: Metriicon 2009
Bugs versus time
Mythical Man Month

![Diagram showing bugs found per month, debugging difficulty, and months since installation. The diagram highlights security and the question of whether security is considered.]
Current Software Engineering Models

The academic community focuses on measuring the quantity of vulnerabilities of software.

More bugs = More Attacks
A Puzzle for you
Even weak software enjoys a Honeymoon
A Honeymoon?

We are defining the Software Honeymoon as the period of time between the first release of a program and the disclosure of its first exploitable vulnerability.

IOW, Honeymoon = Learning curve
Patterns we saw:
Trying to secure software is an Arms Race
The attacker’s curve

We are only beginning to think about individual pieces in the steady state arms race.

- Papers demonstrating how attackers respond to countermeasures
- Papers measuring the rates of infection
- Every “Patch Tuesday” is followed by an “Exploit Wednesday”

But We know almost nothing about the pre-zero day cold war

- Very little research into this area
- We observe that the Honeymoon period is a time of relative peace
An Observation

The Honeymoon ends after attackers:

Research to discover
Exploitable attack vectors
Existence of vulnerabilities
   Soundness of crypto, protocols, implementation, testing
      Complexity is attacker’s friend here
Availability of specs, source code, sample targets
Difficulty of finding the vulnerabilities
   Complexity is actually the enemy of the attacker here

Development
   Build and debug an exploit

Operations
   Find and exploit targets
Security Metrics questions

Our usual question: “Can we measure how secure this system is?”

We analyze the *intrinsic* properties of the system.

A different question: “Can we measure how long will it be before this system is first attacked?”

“What is the expected time to zero-day exploit”

We must model not only the intrinsic security of the system, but the threat and behavior of attackers.

IOW, *extrinsic* properties are at least as important as *intrinsic* properties.
Why this matters?

Intrinsic security properties of software are a poor predictor of when an attack will occur.

Intrinsic security properties of software are a poor indicator of how devastating an attack will be.

Focus on intrinsic security properties leaves us defenseless against new, innovative attacks.

We are spending our attention and resources on the wrong things.

Focus needs to be on extending the Honeymoon.
The Arms Race Today
Improving the Extrinsic Properties of Software

Get a better idea of the length of the Honeymoon period
Develop ways to prolong it
If your honeymoon is over, find ways to jump start it.
May mean completely changing our view of the software life cycle - why?
New code is better than old code.
New code is better than old code, 
*even if it introduces* 
new vulnerabilities

Because, then you get a second 
Honeymoon!