Is finding security holes a good idea?

Eric Rescorla
RTFM, Inc.
Workshop on Economics and Information Security 2004

The facts of life

- All software has bugs
- Security-relevant software has security bugs
- Security is a problem in any multi-user system
  - or any network connected computer
  - ... and almost all machines now are networked
- So what can be done?

Benefits of vulnerability research

- Basic value proposition: pre-emption
  - “Find and fix vulnerabilities before the bad guys do”
- Additional arguments
  - Pressure vendors to fix more quickly
  - Discover new kinds of vulnerability
    - And presumably defenses
  - Provide users with information about quality
    - Thus pressuring vendors to produce higher quality software
- “Openness is good”
- “No security through obscurity”

Costs of vulnerability research

- Attackers get information about vulnerabilities
  - Most attacks are based on known problems
    - E.g., Code Red, Slammer
- Vendors expend effort on developing fixes
- Administrators expend effort on fixing
  - And sometimes the fixes themselves do damage
- The research itself is expensive

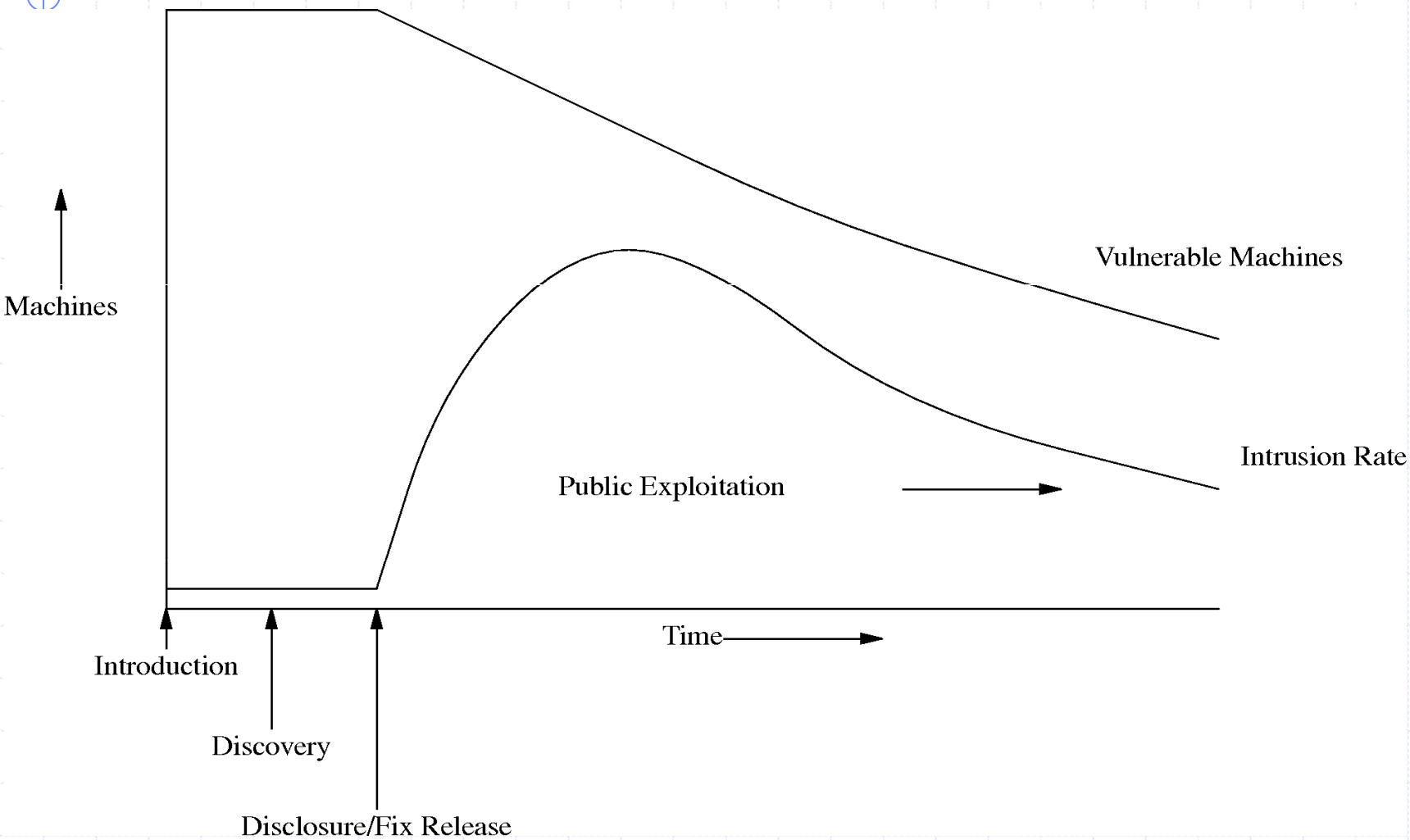
How do we decide what to do?

- Standard of care in medicine
  - First do no harm
  - Impossible to prove a negative
    - There's always some possibility that a treatment works
  - Ask instead if we can show a treatment works
- Right question to ask
  - Can we demonstrate that vulnerability research pays off?
- Data is inherently messy
  - Try to give vulnerability finding the benefit of the doubt

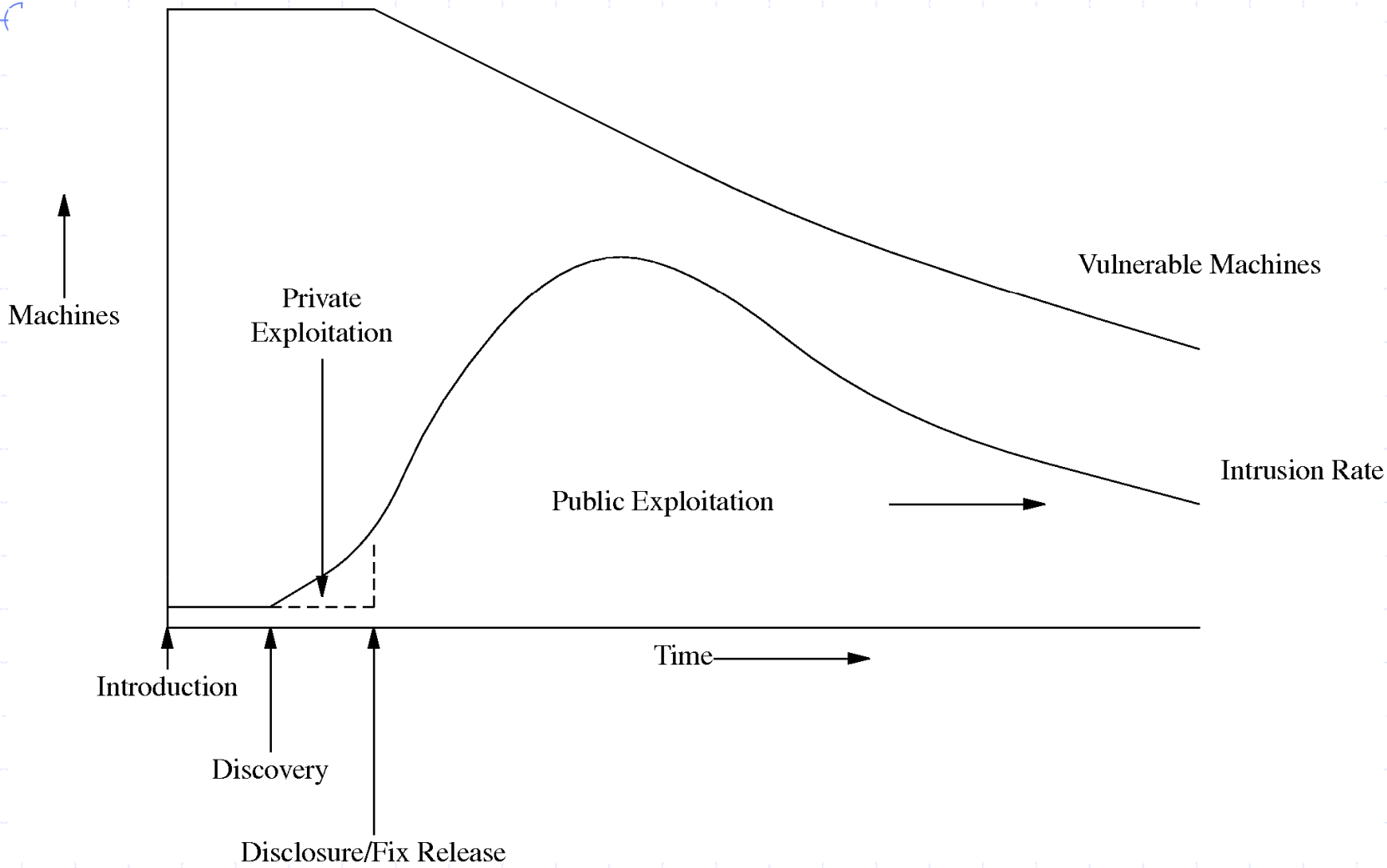
Two discovery scenarios

- White Hat Discovery (WHD)
  - Vulnerability found by a good guy
  - Follows normal disclosure procedure
- Black Hat Discovery (BHD)
  - Vulnerability found by a bad guy
  - Bad guy exploits it

White Hat Discovery Time Course



Black Hat Discovery Time Course



Cost/benefit analysis

- WHD is clearly better than BHD
  - Cost difference
    - $C_{BHD} - C_{WHD} = C_{priv}$
  - If we have a choice between them, choose WHD
- Say we've found a bug
  - Should we disclose?
  - Bug may never be rediscovered
    - Or rediscovered by a white hat
    - ... or discovered much later

Finding the best option

- Probability of rediscovery = p_r
  - Ignore cost of patching
  - Assume that all rediscoveries are BHD (conservative)

	Disclose	Not Disclose
Rediscovered	N/A	$C_{pub} + C_{priv}$
Not Rediscovered	C_{pub}	0
Expected Value	C_{pub}	$p_r(C_{pub} + C_{priv})$

Key question: probability of rediscovery

- Disclosure pays off if $p_r (C_{pub} + C_{priv}) > C_{pub}$
  - Disclosure is good if p_r is high
  - Disclosure is bad if p_r is low
- C_{pub} and C_{priv} are hard to estimate
- But we can try to measure p_r
  - This gives us bounds for values of C_{pub} and C_{priv} for which disclosure is good

A model for p_r

- Assume a program has N vulnerabilities
  - F are eventually found
  - And all bugs are equally likely to be found
    - This is a big assumption and probably not entirely true
- Each bug has an F/N probability of being found
- Say you find a bug b
  - Probability of rediscovery $p_r <= F/N$
- This model is easily extended to be time dependent
  - Assuming we know the rate of discovery as a function of time

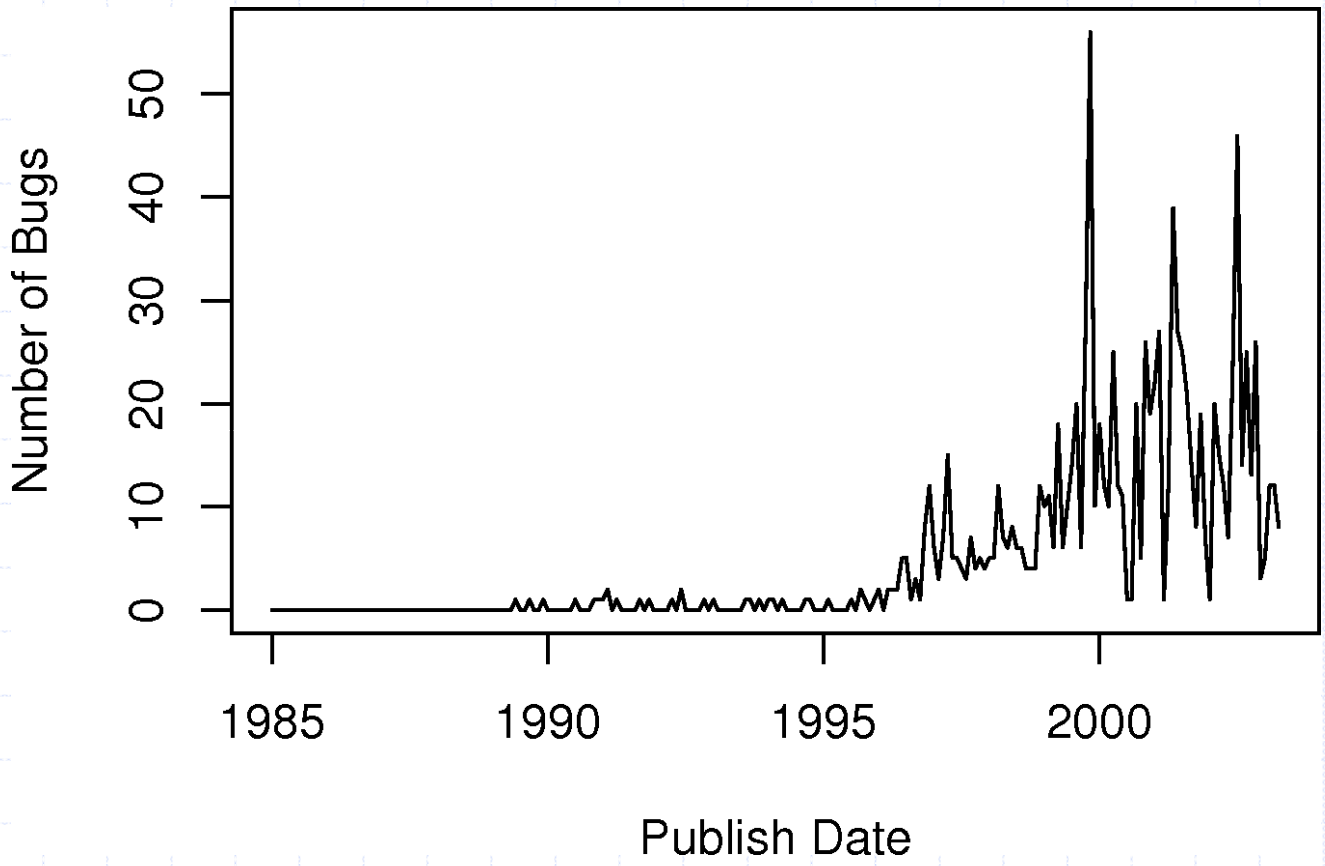
Outline of the experiment

- Collect data on rate of bug discovery
- Use that data to model rate of bug finding
  - Using standard reliability techniques
- Estimate $p_r(t)$

Data source

- NIST ICAT metabase
  - Collects data from multiple vulnerability databases
  - Includes
    - CVE Id
    - affected program/version information
    - Bug release time
  - Used the data through May 2003.
- Need one more data point: introduction time
  - Collected version information for 35 programs

Vulnerability Disclosure by Time



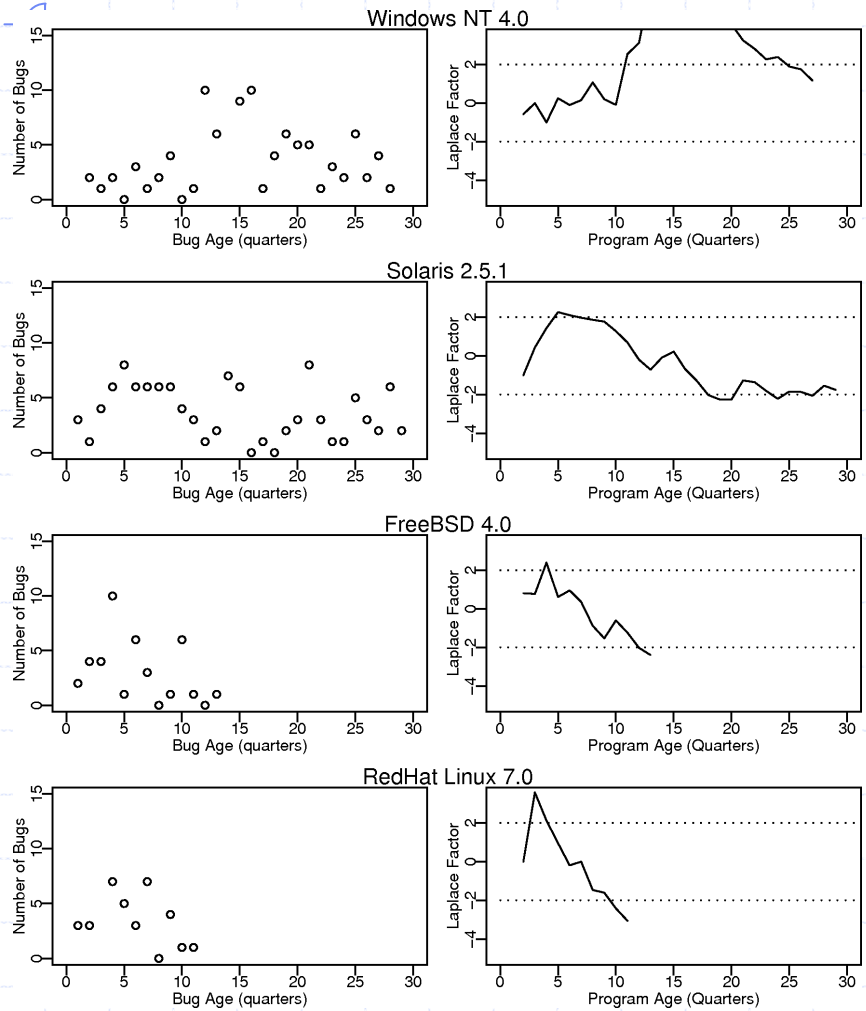
Data issues

- We only know about discovered bugs
  - And have to infer stuff about undiscovered bugs
- Data is heavily censored
  - Right censoring for bugs not yet found
  - Left censoring because not all bugs introduced at same time
- Lots of noise and errors
  - Some of these removed manually

Approach 1: A Program's Eye View

- Question: do programs improve over time?
- Take all bugs in Program/Version X
  - For a few selected program/version pairs
    - Genetically somewhat independent
  - Regardless of when they were introduced
  - Plot discovery rate over time
- Is there a downward trend?

Diclosures over time (selected programs)

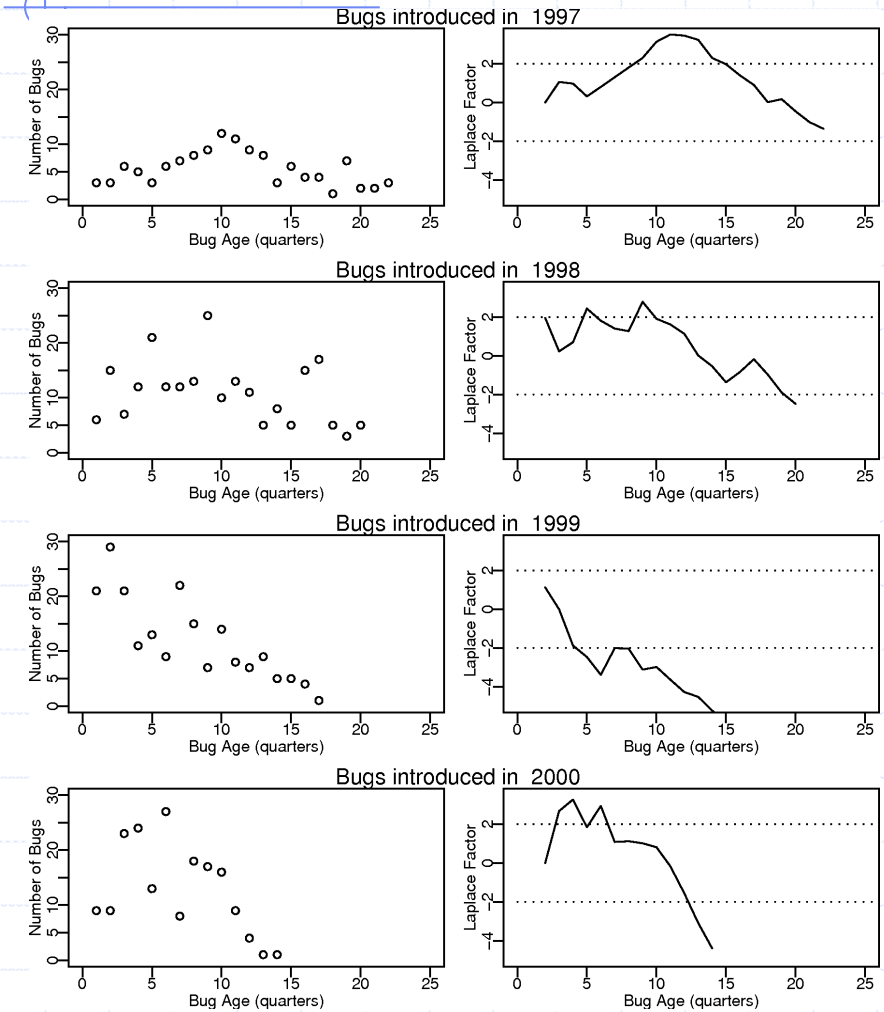


- Linear regression
  - No significant downward trend
- Exponential fit
  - No significant trend
  - Can't even fit Windows
- Laplace factor
  - Only significant depletion at end
    - ... but there are censoring issues

Approach 2: A bug's eye view

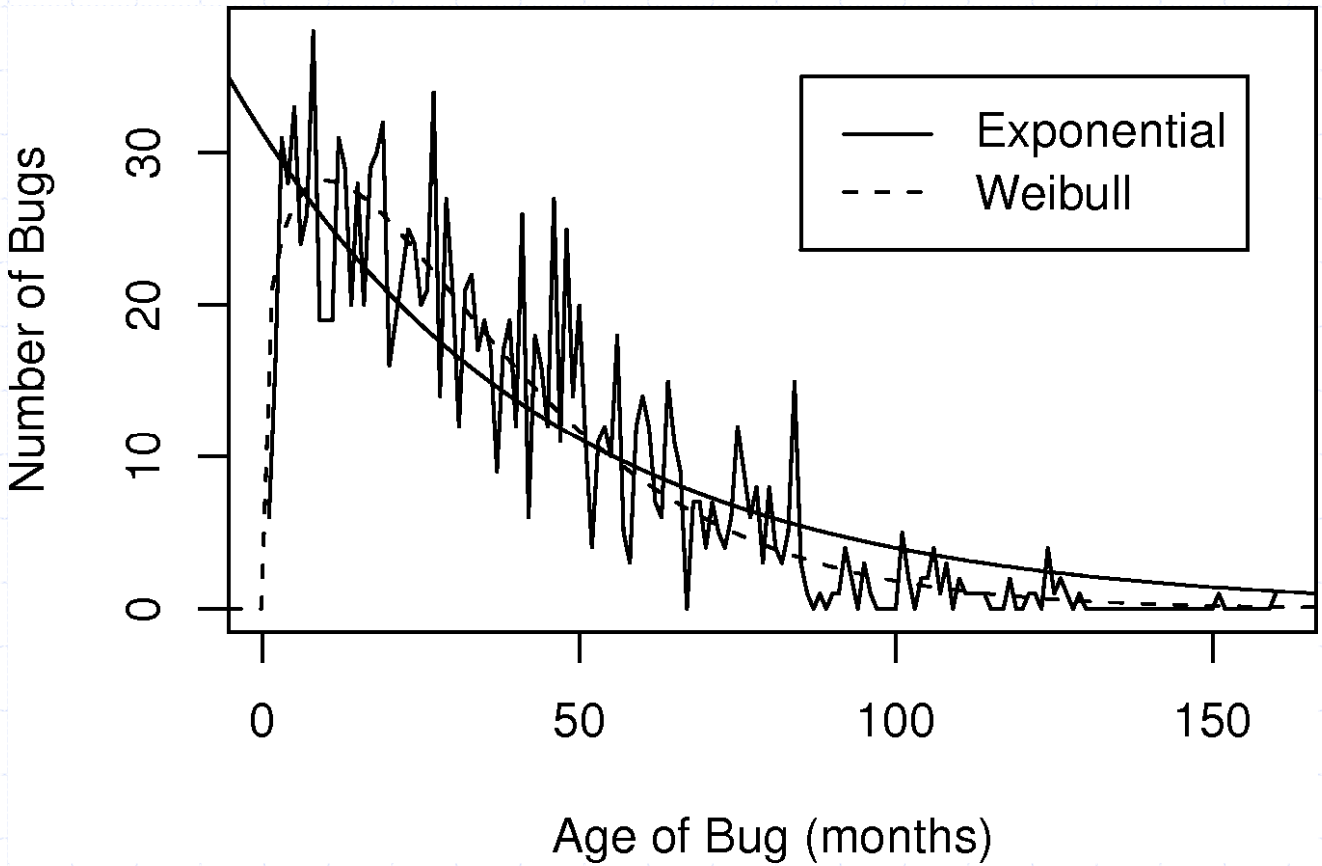
- Find bug introduction time
  - Introduction date of first program with bug
- Measure rate of bug discovery
  - From time of first introduction
  - Look for a trend

Disclosures over time (by introduction year)



- Linear regression
  - Significant trend only for 1999
- Exponential
  - Significant trend only for 1999
- Laplace factor
  - Generally stable

What if we ignore censoring?



Ignoring censoring gives us a trend

- But the improvement in security is very slow
  - Half life is approximately 2.5 years
- One way to look at this: net present value
  - Assume that the bug will eventually be found
    - By a Black Hat
    - But sometime in the future
      - Controlled by p_r
  - Compute the net present value (remembering to discount)
    - .89 C_{BHD} (conservative 3% discount rate)
    - .66 C_{BHD} (aggressive 12% discount rate)
- This gives us a breakeven point for C_{WHD} vs. C_{BHD}
  - Disclosure only worthwhile with a 12-52% margin

Preliminary conclusions

- The evidence of bug depletion is weak
  - It's possible the rate of discovery is constant
  - Or depletion is fairly slow...
- Even if there is depletion, it's not clear bug finding pays off
  - Does the NPV of failing to disclose exceed cost of disclosure
  - ... by enough to justify the cost of research?

Planning for a faster world

- Automatic patching
  - Getting increasingly popular
  - Apparently makes disclosure more attractive
  - But also creates potential for automatic malware
- Faster malware
  - Time to exploit appears to be shrinking
  - Spreading getting faster
  - Makes disclosure less attractive
    - Because patching less effective

Policy Implications: Research

- Deemphasize bug finding
  - It's a questionable activity at best
  - But we don't know it's bad
- Improve record keeping
  - The current data is really noisy
    - It doesn't allow us to draw firm conclusions
  - In particular, what versions are affected
    - This would be good to know in any case
  - Better history of when versions are released

Policy Implications: Disclosure/Fixing

- Fixing without disclosure is better than disclosure
  - This is a clear implication of the model
  - An advantage for Closed Source over Open Source
- It's not clear that fixing is good
  - Especially if you have to disclose
  - Are unknown bugs really dangerous?
    - Of just potentially dangerous
  - Maybe we should do nothing
- What are the vendor's incentives?
  - Disclosure after fixing looks like punishment!
  - Can we charge vendors without revealing bugs?
    - "Buying up loose nukes"…

Review of assumptions

- Biased against vulnerability research
  - Bugs are discovered in random order
- Biased in favor of vulnerability research
  - All bugs are rediscovered
  - All rediscoveries are by Black Hats
  - Ignore data errors that make bugs appear younger
  - Ignore sampling bias introduced by limited study period (when unavoidable)

Conclusions

- Vulnerability finding clearly has costs
  - Research
  - Fixing
  - Intrusions
- Looked for offsetting benefits
  - In the form of reduced bug finding rates
  - Any effect appears to be small at best
- We need more research to know for sure
  - Better data set
  - Prospective study
- Should we be applying treatments we don't know work?