Looking at the big picture on vulnerabilities

Eric Rescorla
Network Resonance, Inc.

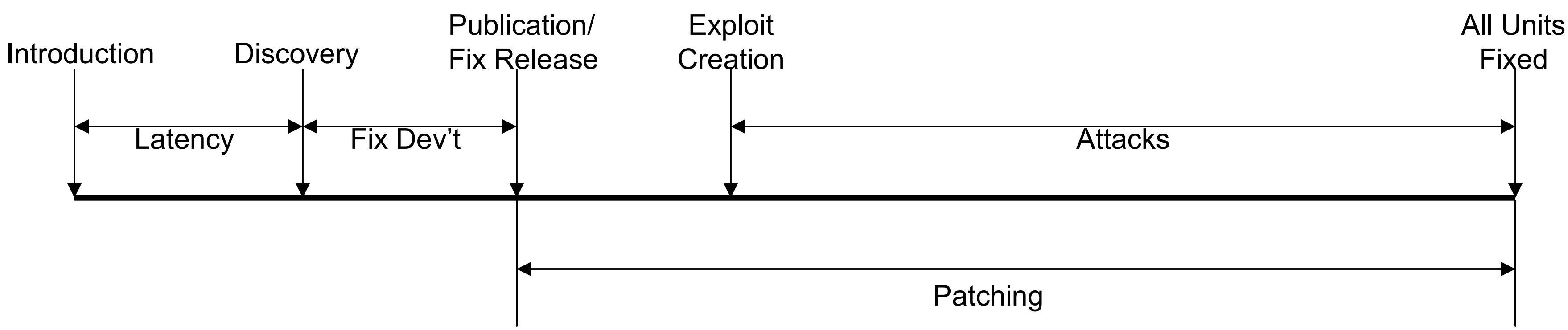
The big picture

- All software has bugs
  - Security relevant software has security bugs
  - Security is a problem in any multi-user system
    - Or any networked computer
    - … and almost all machines now are networked
- How do we choose appropriate behaviors?
  - Policies?
  - Appropriate levels of investment?
- The standard tool is cost/benefit analysis
  - But this requires data

Talk Overview

- Vulnerability life cycle
- Empirical data on discovery
- Empirical data on patching
- Big unknown questions
- Looking beyond computer security

Life cycle of a typical vulnerability



Common Assumptions (I)

- Initial vulnerability count (V_i)
  - Roughly linear in lines of code (l)
  - Some variability due to programmer quality
- Rate of discovery (V_d)
  - Somehow related to code quality
    - $dV_d/dV_i > 0$
    - $d^2V_d/dV_i^2 < 0$???
  - This relationship not well understood
  - Popularity? Attacker tastes? Closed/open source?

Common Assumptions (II)

- Vulnerabilities with exploits (V_e)
  - Some subset of discovered vulnerabilities
    - But who knows what subset?
    - And on what timeframe?
    - Anecdotal evidence indicates it's getting shorter
- Vulnerabilities used in attacks (V_a)
  - Somehow scales with V_e
  - But again how?
  - And what controls the attack rate (A)?
- Loss due to attacks (L)
  - Somehow scales with A

The intuitive model

- Running bad software places you at risk
  - More latent vulnerabilities means more discovered vulnerabilities means more attacks
- Vendor quality improvement reduces risk
  - Converse of above
- Release of vulnerabilities increases risk
  - More attack sites
- But patching decreases risk
  - Less vulnerabilities means less attacks

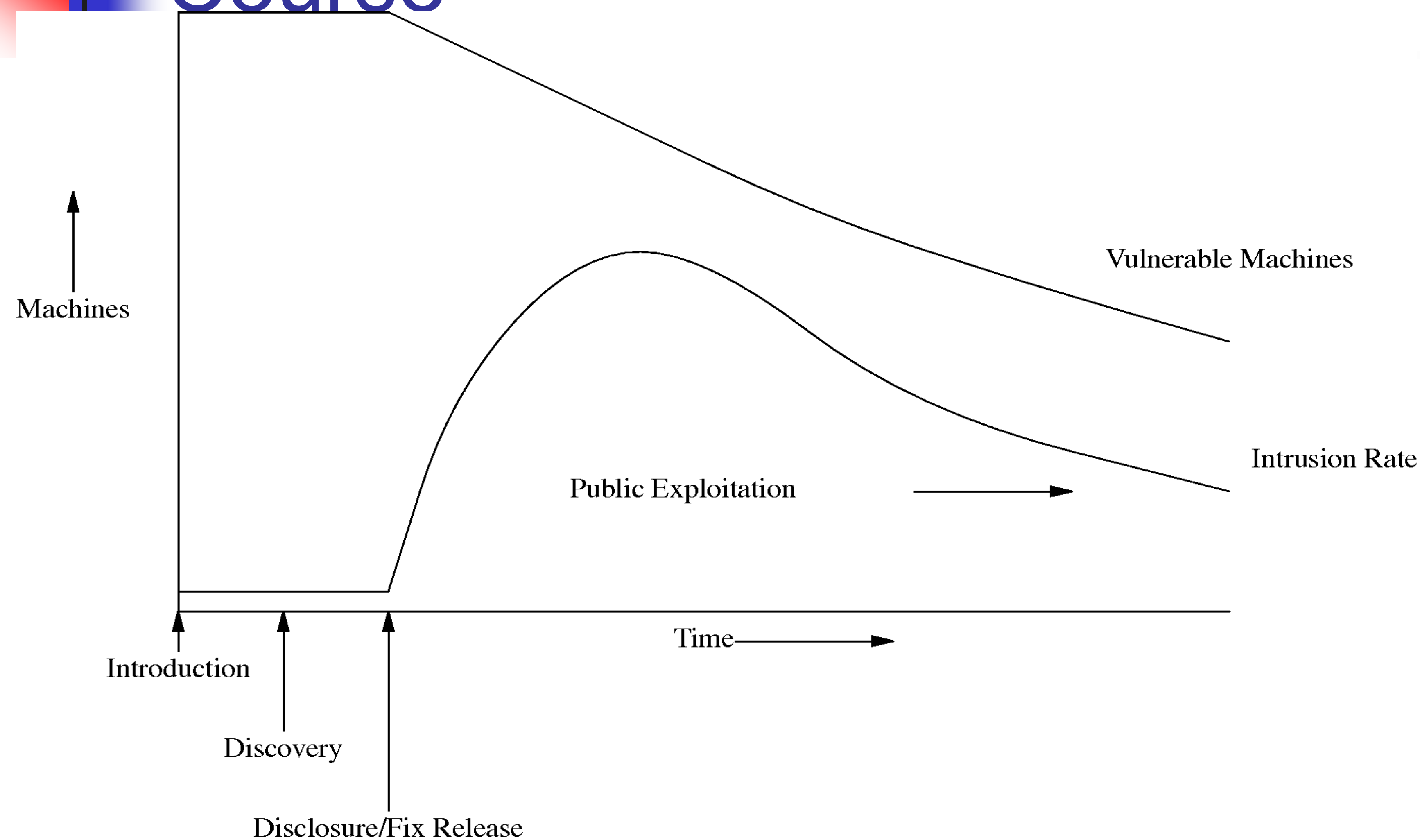
Empirical data on discovery

- Question: is finding vulns socially useful?
- Benefits
  - Pre-emption
    - “Find and fix vulnerabilities before the bad guys do”
  - Incentives for vendors
  - Research
- Costs
  - New tools for attackers
    - Most attacks are based on known problems
  - Cost to develop fixes
    - And costs to install them
  - The research itself is expensive

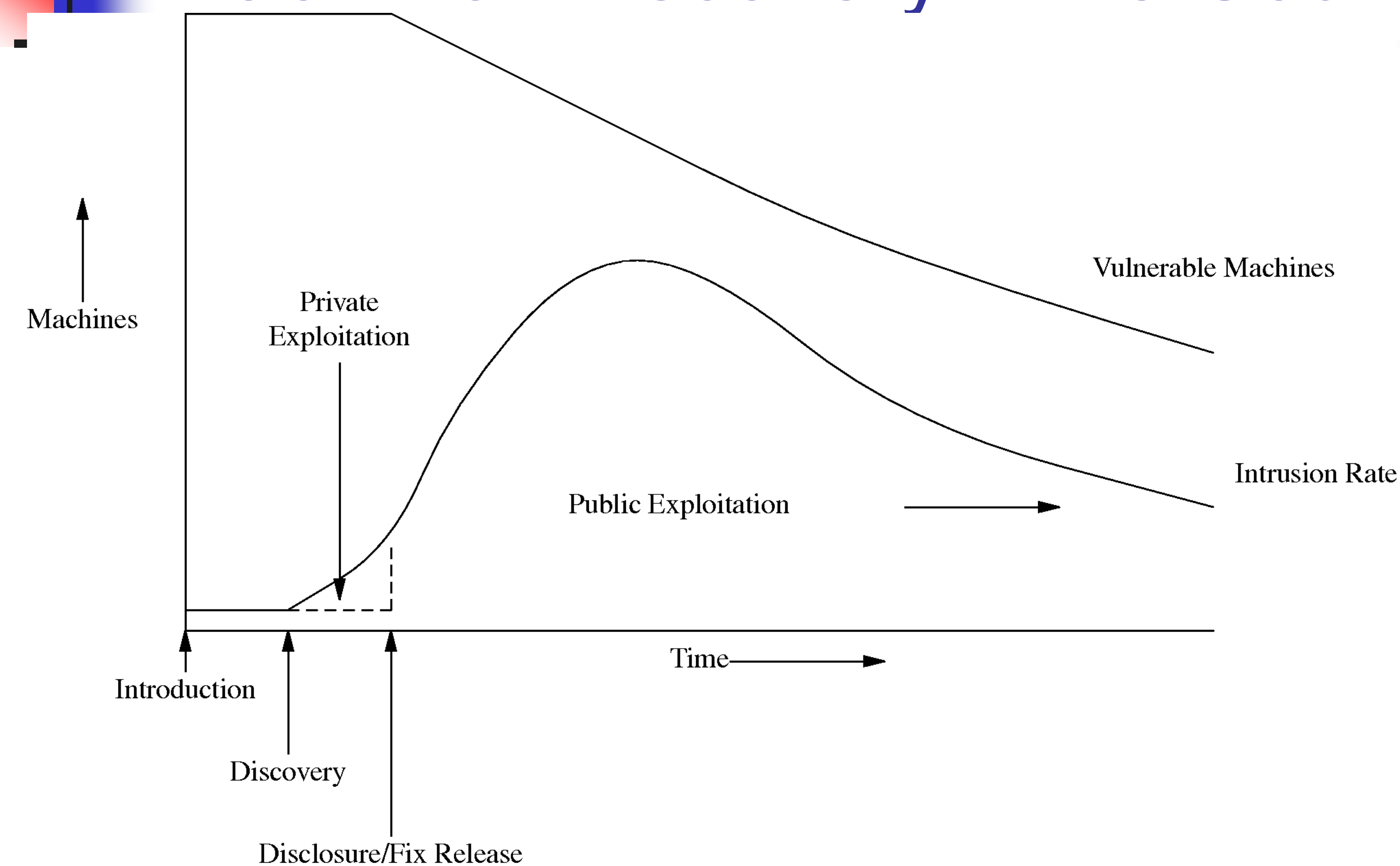
Two discovery scenarios

- 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

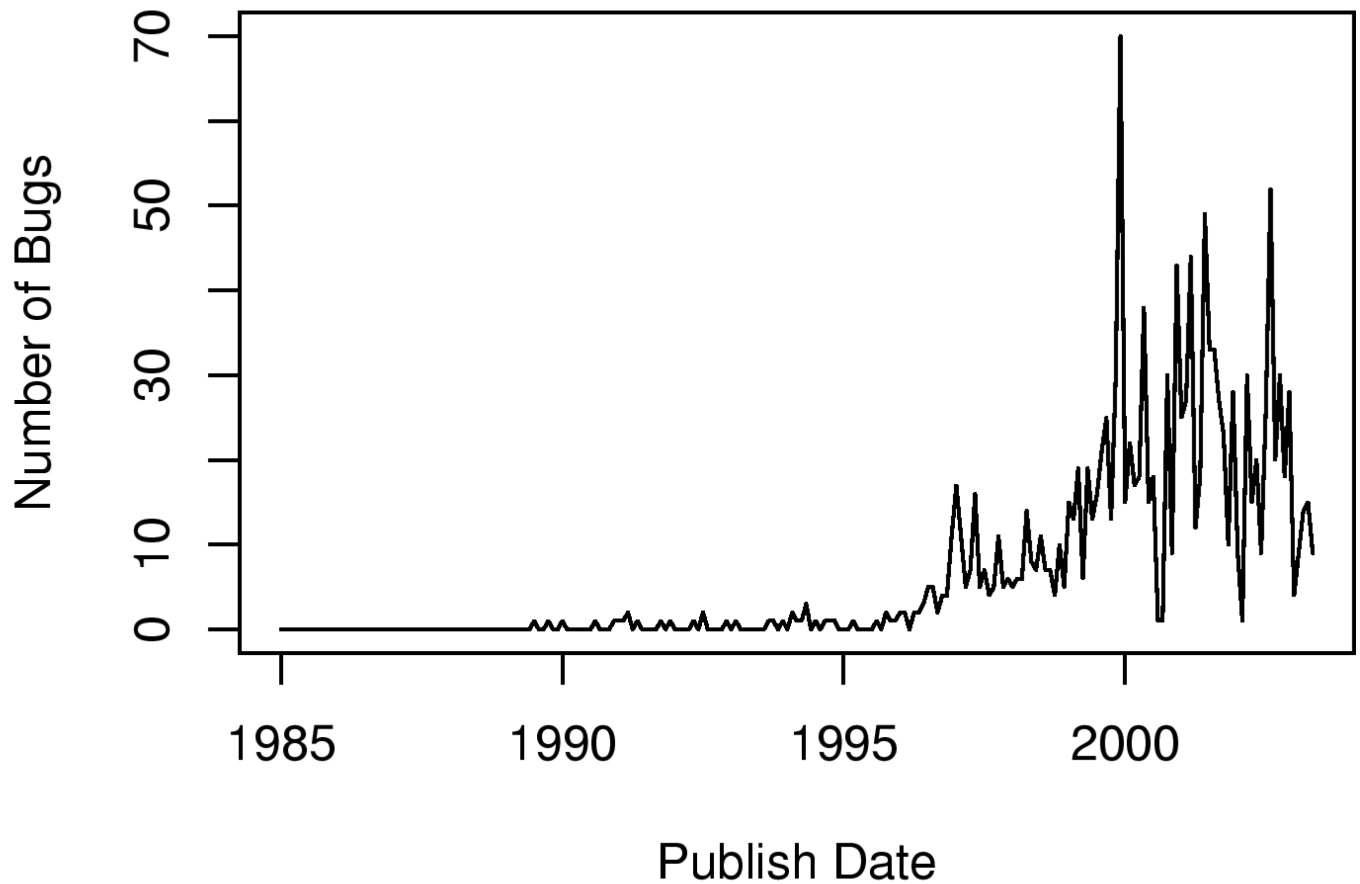
Outline of the experiment

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

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



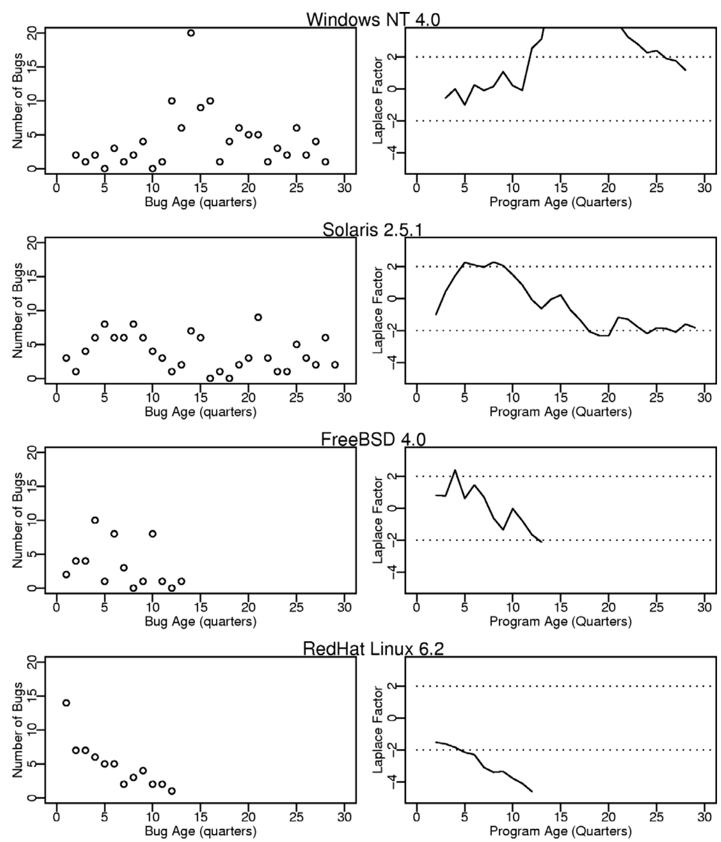
Data issues

- 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?

Disclosures over time (selected programs)

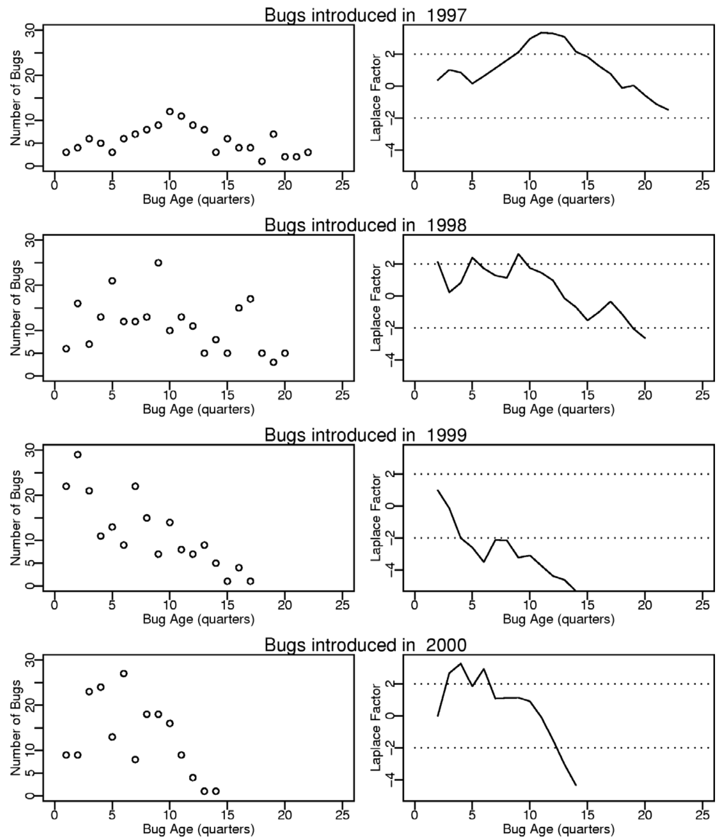


- Linear regression
  - Significant only for RH 6.2
- Exponential regression
  - Significant only for RH 6.2
- Laplace factor
  - Only significant depletion at end (except RH 6.2)
  - ... but there are censoring issues

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

How to think about this

- Medical standard of care
  - First do no harm
  - We’re burning a lot of energy here
  - Would be nice to know that it’s worthwhile
- Answers aren’t confidence inspiring
  - This data isn’t definitive
    - See caveats above
  - Other work in progress [Ozment 05]

Empirical data on patching rate

- Rate of patching controls useful lifetime of an exploit
- So how fast do people actually patch?
- And what predicts when people will patch?

Overview of the bugs

- Announced July 30, 2002 by Ben Laurie
- Buffer overflows in OpenSSL
  - Allowed remote code execution
- Affected software
  - Any OpenSSL-based server which supports SSLv2
    - Essentially everyone leaves SSLv2 on
    - mod_SSL, ApacheSSL, Sendmail/TLS, …
    - Easy to identify such servers
  - Any SSL client that uses OpenSSL

OpenSSL flaws: a good case study

- A serious bug
  - Remotely exploitable buffer overflow
- Affects a security package
  - Crypto people care about security, right?
- In a server
  - Administrators are smarter, right?
- Remotely detectable
  - …easy to study

Questions we want to ask

- What fraction of users deploy fixes?
  - And on what timescale?
- What kind of countermeasures are used?
  - Patches
    - Available for all major versions
    - Often supplied by vendors
  - Upgrades
  - Workarounds
    - Turn off SSLv2
- What factors predict user behavior?

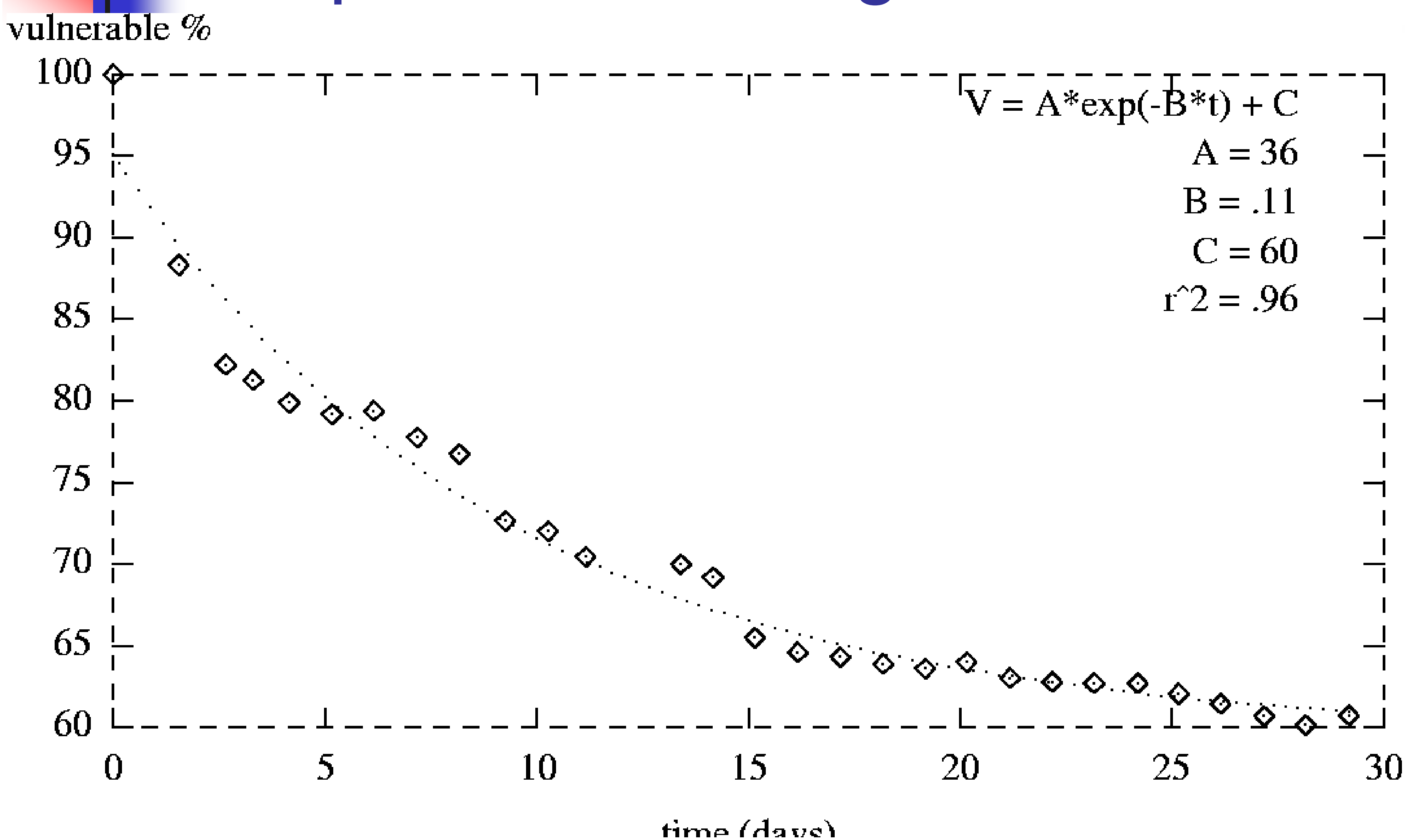
Methodology

- Collect a sample of servers that use OpenSSL
  - Google searches on random words
  - Filter on servers that advertise OpenSSL
    - This means mod_SSL
    - n=892
      - (890 after complaints)
- Periodically monitor them for vulnerability

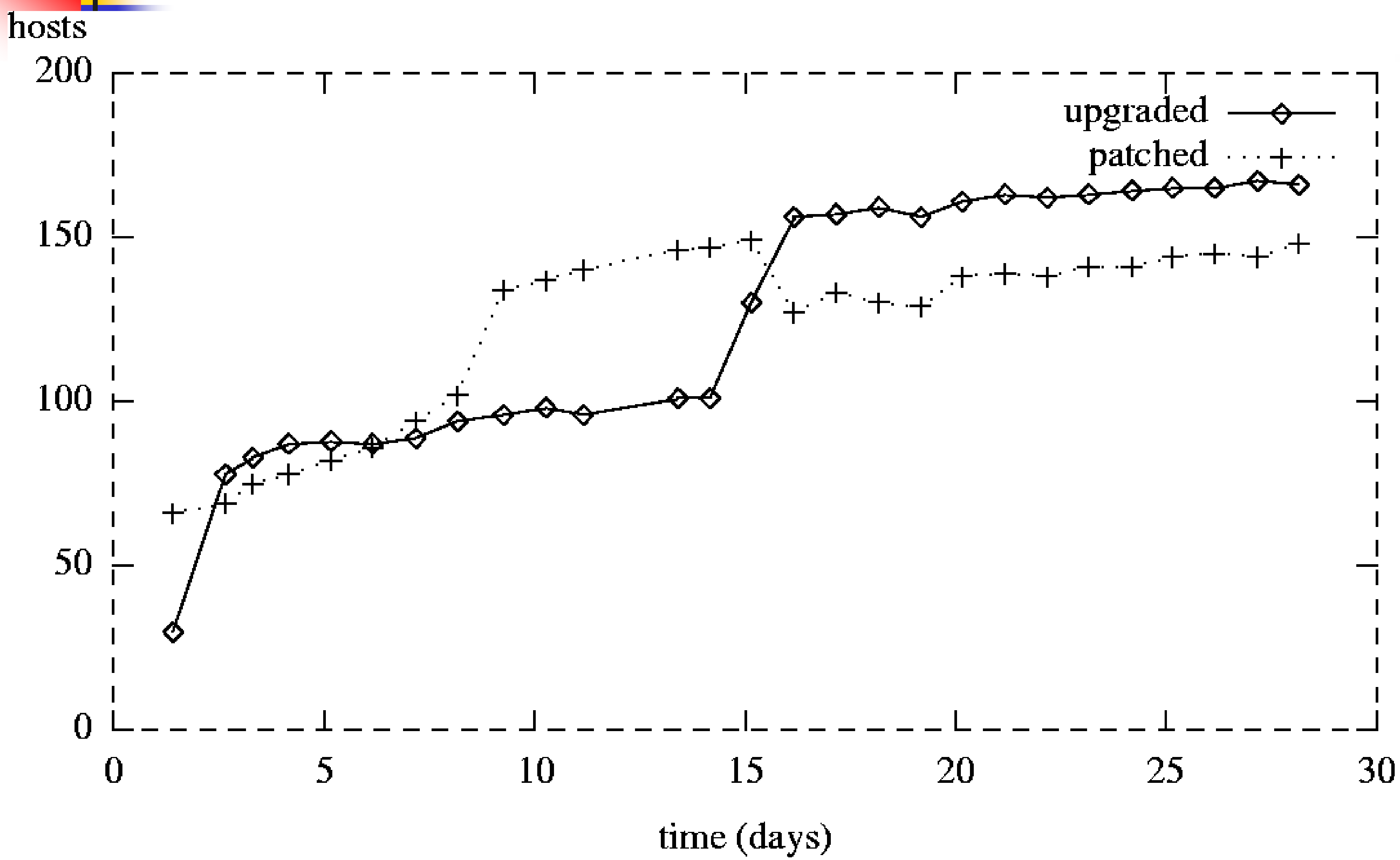
Detecting Vulnerability

- Take advantage of the SSLv2 flaw
  - Buffer overflow in key_arg
- Negotiate SSLv2
- Use an overlong key_arg
  - The overflow damages the next struct field
    - `client_master_key_length`
  - `client_master_key_length` is written before it is read
    - So this is safe
- This probe is harmless but diagnostic
  - Fixed implementations throw an error
  - Broken implementations complete handshake

Response after bug release



Kinds of fixes deployed



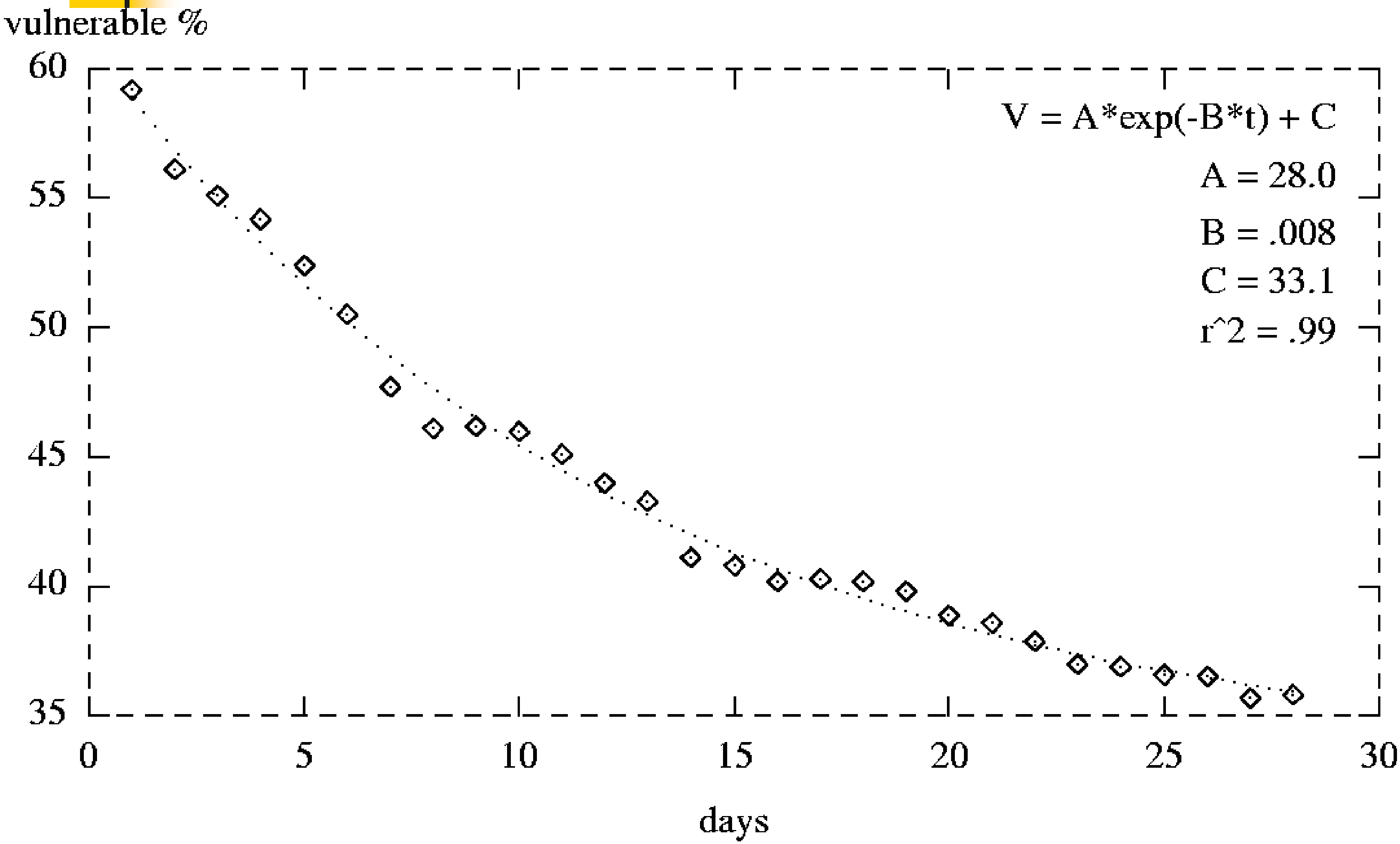
Why not use workarounds?

- Disabling SSLv2 is complete protection
  - It's easy
  - But essentially no administrator did it
    - Never more than 8 machines
  - Why not?
- Guesses…
  - Advisories unclear
    - Not all described SSLv2-disabling as a workaround
    - Some suggested that all OpenSSL-using applications were unsafe
      - It's fine if you just use it for crypto (OpenSSH, etc.)
  - Pretty easy to install patches
    - Anyone smart enough just used fixes

Predictors of responsiveness

- Big hosting service providers fix faster
  - The bigger the better
  - More on the ball? More money on the line?
- People who were already up to date fix faster
  - Signal of active maintenance?
  - Or just higher willingness to upgrade

Response after Slapper release



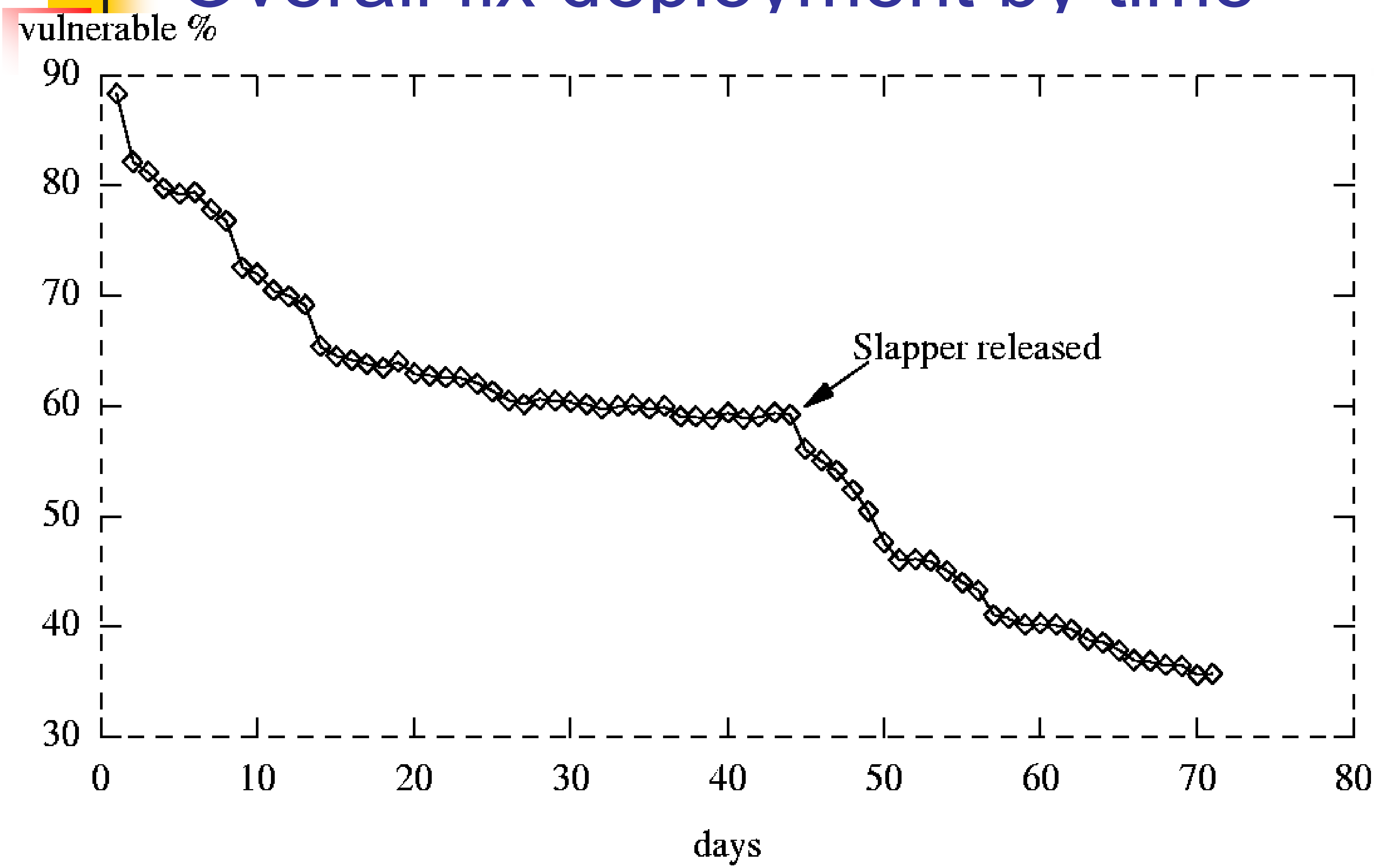
Why so much post-worm response?

- People didn’t hear the first time?
  - Not likely… published through the same channels
- Guesses
  - People are interrupt driven
  - People respond when they feel threatened
    - And deliberately ignore potential threats

The zombie problem

- 60% of servers were vulnerable at worm release
  - These servers can be turned into zombies
  - … and then DDoS other machines
- Independent servers are less responsive
  - So they’re harder to turn off
    - Try contacting hundreds of administrators
- Slapper wasn’t so bad
  - Since Linux/Apache isn’t a monoculture
  - And the worm was kind of clumsy

Overall fix deployment by time



Have things changed?

- Automatic patching more common
- Threat environment more hostile
- Answer: not much [Eschelbeck ‘05]
  - Externally-visible systems: half-life = 19 days (30 days in 2003)
  - Internally-visible systems: half-life = 48 days (62 days in 2005)

The big open question

- How much do vulnerabilities cost us?
- How much would various defenses cost us?
- How well do they work?
- Where should we be spending our money?

Steps along the way

- What are the predictors of discovered vulnerability rate?
  - Software quality? Popularity?Access to source code? Hacker attitudes?
- What is the marginal impact of a new vulnerability?
  - Number of total attacks?
  - Cost of attacks?
- What is the marginal impact of faster patching?
  - How much does it reduce risk?
  - Balanced against patch quality [Beattie ‘02, Rescorla ‘03]
- Does diversity really work?
  - Targeted vs. untargeted attacks
  - What about bad diversity [Shacham ‘04]

Thinking outside CS: Bioweapons

- Same as software but with much worse parameters
- Vulnerabilities are long-standing
- Exploits are hard to create
  - But there are plenty of old ones available
    - Smallpox, anthrax, ebola, etc.
  - And technology is making it easier
- Fixes are hard to create
  - Where's my HIV vaccine?
  - And easy to counter
    - Influenza
    - Mousepox [Jackson '01]
- Patching is painfully slow

Case study: 1918 Influenza

- Complete sequence has been reconstructed
  - Published in Science and Nature ‘05
  - Includes diffs from ordinary flu
    - And explanations
- Usual controversy [Joy and Kurzweil ‘05]
  - But what’s the marginal cost?
  - Smallpox has already been fully sequenced
    - Current vaccination levels are low
      - And vaccine has bad side effects
      - And compare the mousepox work
    - Possible to de novo synthesize the virus?
  - What’s the impact of a new virus?

Final slide

- Questions? Interested in working on this stuff?
- Reach me at ekr@rtfm.com