# Bug Auctions: Vulnerability Markets Reconsidered

Andy Ozment

14 May 2004
Workshop on Economics and Information Security
Minneapolis, MN, USA

# Motivation

- No good way to measure software security
- Software market as a 'market of lemons' (Anderson 2001)
- Solution: Vulnerability Market (VM)
  - Establish a product's cost to break
  - (Schechter 2002a, 2002b, 2004)

# A Simple Vulnerability Market

Software *producer*

- Pays a reward, $R$, for each unique vulnerability reported
- Offers reward in both pre- and post-release phases
- Makes software freely available to testers (*)
- For closed source software, testers get executables

*Tester*

- Report vulnerabilities
- Anybody can be a tester

* Addressed in more detail in the paper.

# Producer's Motivation

What Value Does the Producer Obtain from the VM?

- Improved product quality
  - Attract larger fraction of pool of existing testers
  - Grow the pool
- Useful metric
  - Relative metric to differentiate its product
  - Metric to judge the quality of outsourced coding
  - For producer that has invested effort in secure coding

Assumes vulnerabilities are ordered.

- Shared vulnerability finding heuristics
- The same vulnerability will be found more than once

# Product Differentiation

The VM Allows the Producer to Differentiate Its Product

- $R$ is the *lower bound* on its product's cost to break
- Establish *upper bound* on competitor's product's cost to break
  - Buy a vulnerability for the competitor's product
  - Use a trusted third party
  - Pay $< R$
- The producer's product is thus more secure

# Software Producer's Criterion

But can we improve the idea?

The producer is interested in:

- Value: Pay as little as possible for each vulnerability
- Speed: Fix vulnerabilities as early as possible
- Order: Fix easy-to-find vulnerabilities first

# A More Advanced Vulnerability Market

Pre-release

- Use a continuously increasing reward
  - $R = R_0 + tr$
  - $R_0$ is minimum reward
  - $t$ is the time since the reward was last claimed.
  - $R$ reset after each report
- Maximizes value at the expense of speed

Post-release

- Either use a continuously increasing reward
  - $R_0$ is the reasonably high minimum security assurance
  - the security assurance increases until reward reset
- Or use a constant reward

# Additional Complexities

- Use VM to find quality defects
  - pre-release phase only
  - $R = (R_0 + tr)x(\text{severity})$
  - reset $R$ even for minor defects
- Trusted third party
  - assesses bug value
  - assesses uniqueness
  - tests reports $\rightarrow$ reports contain exploit
  - ensure tester pseudonymity (*)

# The Vulnerability Market as a Bug Auction

- Ascending first-price (reverse Dutch)
- Open
- Sequential multi-item
- Variable demand
- High entry costs
- Negligible bid costs
- Bidders
  - independent
  - private value
  - asymmetric (*)

# The Format of the Bug Auction

Bidders are asymmetric → this auction is not revenue equivalent.

Is the reverse Dutch format most appropriate for a bug auction?

Conditionally, yes:

- It conveys no information about the number of bidders
- It is preferred by risk averse producers
- A reward is always offered

# Endogenous Entry

Producers want to encourage testers to enter the auction.

- Value of 1 extra bidder $\gg$ profit from res. prices or entry fees

Enhancement 1: Employ a large $R_0$ for first auction(s) in sequence

- Or increase reward more rapidly / use discontinuous jumps
- After initial auction(s) keep $R_0$ reasonably high
- Benefit
  - jumpstarts sequence
  - increases speed at some cost to value
  - help testers amortize cost of learning product

# Variable Demand

Two types of demand uncertainty

- Bug uniqueness → lower prices (good)
- Sequence length → less participation (bad)

Enhancement 3: Reduce length uncertainty to induce participation

- Producer commits to minimum duration/budget
- Auction ends when
  - minimum budget consumed
  - or chronological cap surpassed
  - (whichever occurs first)

# Bidder Attacks: Resale

Resale

- Bidder reports vuln, then resells it on black market
- Bug auction could thus decrease security
- This problem cannot entirely be resolved

Enhancement 4: Reduce reward if exploit is found in the wild.

# Practical Auction Design

Most important aspects of practical auction design

- Encouraging entry
- Preventing collusion
- (Klemperer 2004b)

# Bidder Attacks: Collusion

Enhancement 5: Do not reveal the exact number of testers

- Make collusion impractical
- Prevent colluders from punishing defectors
  - unknown number of testers
  - pseudoanonymity

Does not solve engineer-tester collusion

- Engineer could intentionally plant bugs
- Defenses
  - reward engineers on quality
  - sting operations
  - existing legal/institutional tools
- No great solution, but not a new problem

# Producer Attacks

Submitting a (planted/held in reserve) bug to

- Reset $R$ if it is large
- End auction sequence early

Enhancement 7: promised funds held in escrow with TTP

Additional disincentives against these attacks

- Large unclaimed reward is good PR
- Sequence has chronological cap, just wait
- Accounting/legal obstacles to reclaiming funds
- Reputation damage from reward being claimed

# Fundamental Problems

- First to market $\longrightarrow$ minimal pre-release auction
- Potential cost of assurance
- Flurry of bugs when product released
- Reality vs perception of security
- Pay for testing that used to be free
- Copyright infringement
- Are vulnerabilities found in order?

# Conclusion

Vulnerability Market & Bug Auction

- Provide a comparative measure → stop the lemons effect
- Improve pre-release testing
- VM better understood as an auction
  - Endogenous entry
  - Variable demand
  - Collusion
  - Resale
- Assumes vulnerabilities are ordered → will be re-found

Innovative solution to a vexing problem.
Would require significant cultural / business practice changes,
but the potential gains seem worth the investment.