



# Embargoing the Open

Challenges for Temporary Secrecy in Open-Source

Red Hat Product Security

# Product Security Overview



Red Hat Product Security provides the guidance, stability and security needed to confidently deploy enterprise solutions.

Red Hat Product Security ensures Red Hat products are secured by

- Identifying security issues
- Assessing the severity
- Creating updates
- Notifying customers
- Distributing updates

# About Us



**Fábio Olivé** Vulnerability Response Manager | Red Hat Product Security

*He does actual work*



**CRob** Sr. Program Manager | Red Hat Product Security

*He draws circles and boxes!*

# Embargoes

Embargoed issues are those that, due to their severe impact, are shared on a strictly need-to-know basis, between reporter and vendors, so that they have a chance to prepare updates before the flaw goes public, minimizing the impact on the end-users.

Red Hat prefers to work with short embargo periods (few weeks), although we fully respect longer embargoes owned by external entities.

Wide open development processes pose interesting challenges here.

# THANKS FOR COMING!

Enjoy the rest of the conference!

Just kidding.....

(maybe)

# Culture Shock

RH prides itself on transparency and openness....  
intentionally hiding something is fairly difficult for our  
employees, partners, and customers to take from us.

# And the REALLY fun thing....

Frequently the people that need to work on the flaw aren't even Red Hat employees.

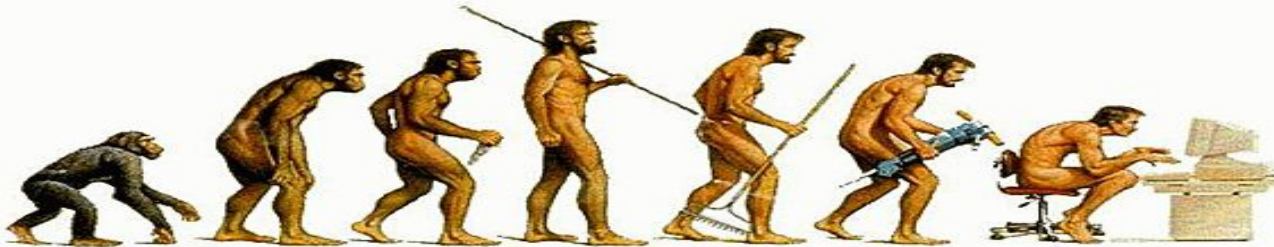
Navigating the upstream seas is challenging by itself, let alone keeping volatile secrets.



# A (Brief) Review of Historical Industry Disclosure Behaviour

# The Stone Age

- Most companies didn't always understand security
- Some would threaten reporters of flaws
- Many relied on the code being a black box; Could somewhat easily deny flaws
- Was easy to keep embargoes, though, sometimes forever ;-)



# The Modern Age

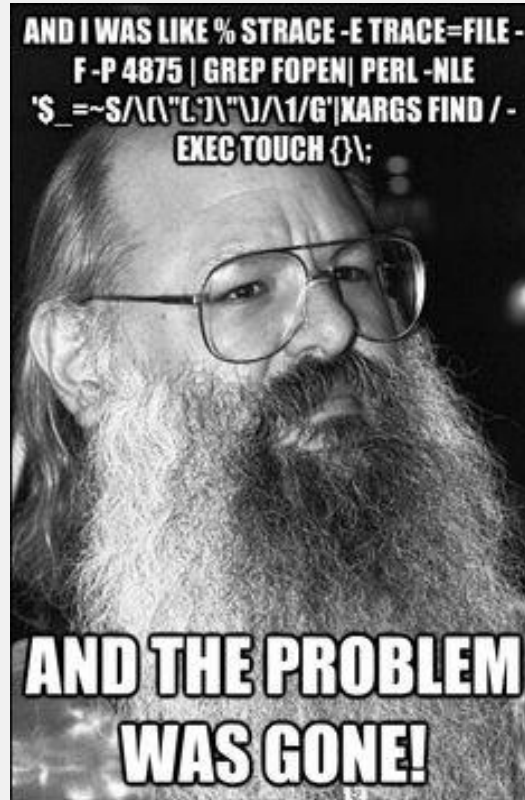


- Security through obscurity is dead
- Companies now mostly grok security
- Have security contacts and disclosure processes
- Bug bounties versus the 0day market
- The Internet forces companies to acknowledge flaws; so embargoes must be kept to reasonable periods

# How Open Source Differs

- Projects did tarball releases
- People would email authors with patches
- Emphasis on free code drops, no provision for an open development process
- Very easy to keep embargoes

*Old School*



- Not just open code, but open process
- Major win for IT and innovation
- Many people contributing to the solution
- **Makes embargoes very tricky, though**

**NEW SCHOOL**

# How Red Hat Works (ish)

# From Pet Project to Enterprise-class

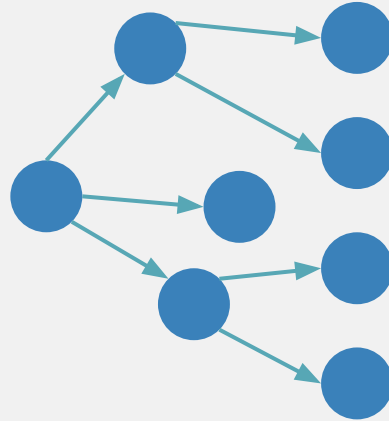


**CHUCK NORRIS DOESN'T WRITE CODE**

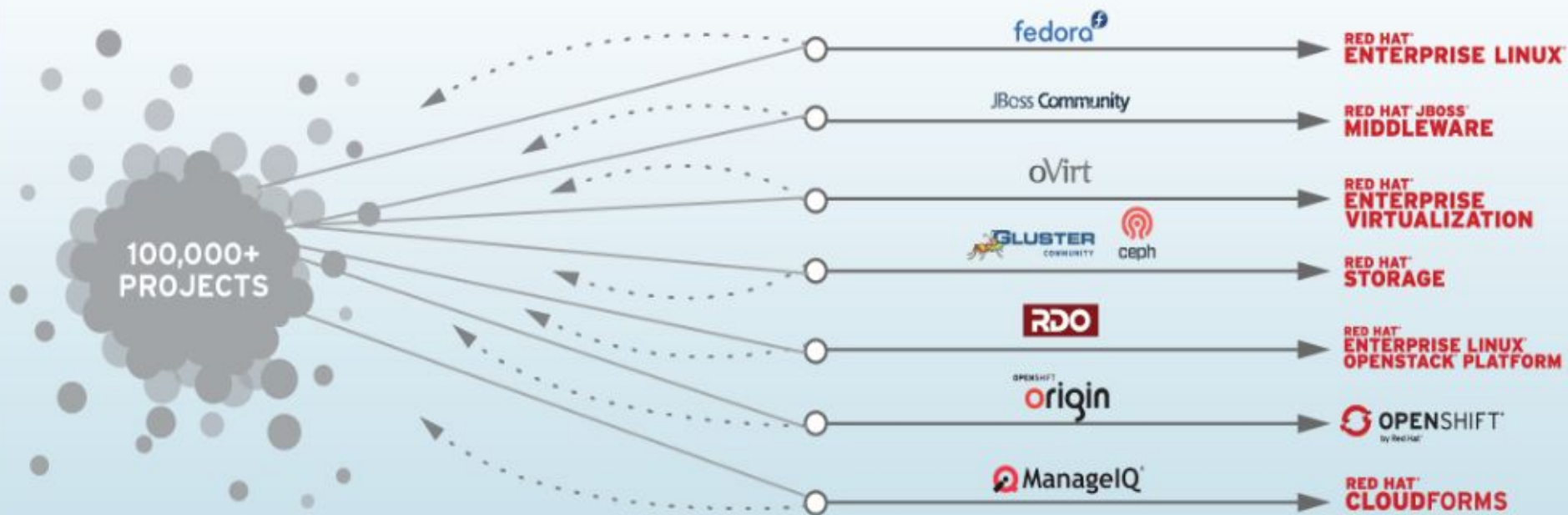
He stares at a computer screen until he gets the program he wants.

# Trusting Your Supply Chain

- Where does it come from?
- Who is your 3rd party's 3rd party?
- Can you trust it?



# DEVELOPMENT MODEL





# Kinds of upstream projects

## Good

Have security contacts, ability to coordinate

## Bad

Dead projects, for-fun projects

## Ugly

Silent fixes, no regard for Security

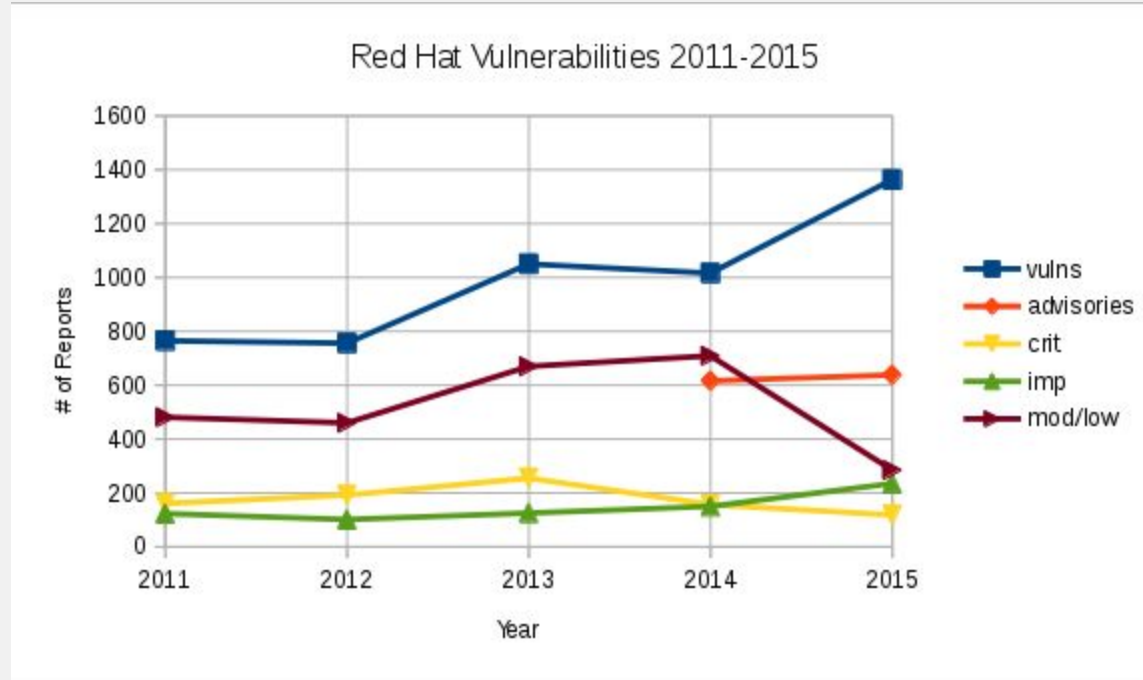


And our customers expect the EXACT same experience for all of them!

# An Overview of Vulnerabilities over time

## Vision Statement

To help protect customers from meaningful security concerns when using Red Hat products.



Year	<u>vulns</u>	advisories	<u>crit</u>	imp	mod/low
2011	765		161	123	481
2012	756		192	101	460
2013	1050		255	125	670
2014	1016	616	157	150	709
2015	1363	638	118	235	285

# 2015 view - Red Hat Vulnerability Sources

# Issues	Percentage	Source	Advance?
8	.6%	CERT	Y
49	3.6%	CVE	N
22	1.6%	Distros	Y
14	1%	Individual	N
51	3.8%	Individual	Y
808	59.3%	Mailing List	N
36	2.6%	Other Vendor	N
167	12.2%	Red Hat	Y
202	14.8%	Relationship	Y

We get advanced notice on ~33% of vulnerabilities

# Open Source Tensions on Embargos

# The **code** is public...

- The foremost tension is that the code, and thus the flaws, are public and wide open ...Not everyone notices the flaws, thankfully.
- We have to assume that if someone reports a flaw, others could also know about it.
- High sense of urgency, can't "sit" on issues.

## ...The **process** is wide open

- How can one prepare an update in private, using public/open infrastructure?
- Can open projects maintain private infra "for the Greater Good" of handling Security issues?

# Upstreams may be dead


- Projects may lose steam over the years
- Forks cause projects to lose developers
- What if the last commit on a project you use was 10 years ago by someone who is now retired?



# Other Upstream Concerns



- One or a few developers doing it for fun; generally they cannot provide adequate security response or coordination
- Lack of security training and understanding of impact of unintended consequences of changes
- Could still be providing awesome innovation, though!
- To Fork or not to Fork....



**WHAT IF I TOLD YOU**

**"THREAT INTELLIGENCE"  
IS JUST A TWITTER FEED**



# Is it a Bug or a Vulnerability?

- Some upstreams treat flaws and bugs equally
- Near equivalent to Full Disclosure (without thought about downstream impact)
- Resistance to requesting CVEs



# “Silent fixes”



- Some projects will just go fix something in their public project, without sufficient information given to understand the security implications (downplaying severity).
- Commit now, “release” later - We can’t break the embargo [it’s not our embargo]

# “Pure” community distributions

- Hard to demand secrecy from volunteers
- Hard to manage things like encryption keys
- Lack of private infrastructure
- Takes time to earn trust from other players



# What We've Learned So Far...

(and continue to everyday...)



**KEEP CALM  
AND  
SHARE THREATS**

# It's not all bad



The majority of people out there are genuinely trying to do the Right Thing.

There's a lot of goodwill and passion inherent to Open-Source.

# Short embargo periods work best



- Usually 2 weeks of embargo time works great!
  - That's our average
  - Large enough to research and test; Short enough to keep focus
- When the unembargo date comes, don't hold back.
- Give as much clear and accurate information as possible, enabling others to evaluate their own risk, as well as fend off media hype.

# Bugfix versus Mitigation

- Providing mitigation steps is almost as important (sometimes more important) as providing the actual bugfix.
- Many users will take a long time to update, but can change configurations easily.





# Vendors: Work with and Help upstreams

- Be the older, experienced brother
- Help tiny upstreams coordinate disclosure
- Coordinate with other vendors to adopt dead upstreams



# Upstreams: Some advice

- Have a clearly named security contact or team, and guidelines to get in touch - Most researchers will happily abide
- Have a simple page up with changelogs, or some other mechanism like RSS feeds
- There is a FIRST SIG working on best practices for this, get involved with it

# Closing Thoughts

- Treat researchers with respect (even the small guys might have something of value to share to help make **you** better).
- Researchers are combing over ALL patches to see what we're doing and if we're doing what we said we did.
- Need to be faster (agile) with delivery of updates.
- Point is not to ruin the fun of coding OSS, but to instruct, guide and teach so they continue innovating.
- When in doubt, talk to oss-sec!



“Giving calm, timely, and accurate information is best” - Cliff



