Xen Project Contributor Training
Part 4 : Culture

Lars Kurth
Community Manager, Xen Project
Chairman, Xen Project Advisory Board
Director, Open Source Business Office, Citrix
Content

Theory: Open Source Flywheel

The demands on what vendors and users want from Xen Project is changing using the Flywheel to illustrate

The project has a recent history of change

Example: The history of the Security Vulnerability Management Process

Other examples of recent and ongoing changes

New demands on the project: New Features/Community Growth vs. Review Process and Review Capacity

New demands on the project: New Features/Community Growth vs. Quality and Security

Feature Lifecycle Management and Documentation
Theory:
Open Source Flywheel
Community Growth

- More Users
  - More business opportunities and momentum
- Open Source Development Model
- Better Product and Experience
  - Lower deployment cost and risk
- More Development Activity
  - Lower development cost
Community Growth

More Users
More business opportunities and momentum

Open Source Development Model

Better Product and Experience
Lower deployment cost and risk

More Development Activity
Lower development cost
Community Growth

More Users
More business opportunities and momentum

Better Open Source Development Model
More efficiency and innovation

Better Product and Experience
Lower deployment cost and risk

More Development Activity
Lower development cost
War Stories:
Tragedy of the Commons
(sort of)
So what happened and why?
OpenSSL Stats

Prior to Heartbleed
Growing Codebase
Static and small contributor base
1 person maintaining 100 KLoC = Underinvestment
Extremely large user base
Critical infrastructure component
Thus impact of Heartbleed is huge

Large user base did not translate into developer community growth

Source: Ohloh.net
Imbalanced Cycle

- Users
- Open Source Development Model
- Product and Experience
- Development Activity
Lesson for Xen Project

Stay vigilant to sustain a balanced Flywheel
Drivers for Change

The Demands on what vendors and users want from Xen Project is changing
Little scrutiny by the tech press
Mostly happy
Fairly disengaged

Established and stable development model

Features
Performance/Scalability
Quality

Lower development cost
Community Growth

2014 and before
Huge amount of scrutiny by the tech press (security, process, releases, …)
Some users unhappy (status quo vs. change)
Vocal users and vendors (the odd “rant”)

2014, 2015, Future …

Community is forced to change:

Features
Performance/Scalability
Higher Quality
Security
Usability / Integrations

More competition (e.g. Containers, Docker, …)

Lower development cost
Community Growth (not at all cost)
New Players: Security, Embedded, …
New Regions: e.g. China & Ukraine
More aggressive product roadmaps
Xen has a history of recent change

External factors are accelerating the amount of change
Example:

Evolution of Xen Project Security Vulnerability Process

xenproject.org/security-policy.html
**V1.0 : Modelled on Debian**

Goals:
Allow fixing, packaging and testing;
Allow service providers to prepare (but not deploy) during embargo

Pre-disclosure:
Membership biased towards distros & large service providers
No predefined disclosure time
July 2012: CVE-2012-0217, Intel SYSRET
Affected FreeBSD, NetBSD, Solaris, Xen and Microsoft Windows

A large pre-disclosure list member put pressure on key members of the Xen Project Community to get an embargo extension
They eventually convinced the discoverer to request an extension
Community Consultation to improve our process

Centered on:

Predetermined disclosure schedule: 1 week to fix, 2 weeks embargo

Who should be allowed on the pre-disclosure list
Fairness issues between small and large service providers
Direct vs. indirect Xen consumers
The risk of larger pre-disclosure list membership
V2.0: Clarifications

Strongly recommended disclosure schedule
Inclusive pre-disclosure list membership
Changes to application procedure (based on checkable criteria)
Sept 2014: [CVE-2014-7118](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-7118)
Leading to the first Cloud Reboot

AWS pre-announced cloud reboot to their customers
Other vendors didn’t.
Policy was interpreted differently by vendors.

This highlighted *ambiguities* in the project’s security policy
(what can/can’t be said/done during an embargo)
V3.0: Deploy & Optimizations

Goals:
Allow fixing, packaging and testing
Allow service providers to prepare (and normally to deploy) during embargo

Pre-disclosure:
Clearer application criteria
Public application process (transparency)
Clear information on what is/is not allowed during an embargo (per XSA)
Means for pre-disclosure list members to collaborate
May 2015: **CVE-2015-3456**
First time we were affected by a branded bug
QEMU bug, which was handled by several security teams: QEMU, OSS Distro Security, Oracle Security & Xen Project
From a process perspective: were not able to provide a fix 2 weeks before the embargo date ended

Conducted **XSA-133 Retrospective** upon request
Process change: Earlier embargoed pre-disclosure without patches
Examples:

Of other recent changes
And changes under discussion
## Other Changes in the last 2 years

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design Reviews</strong></td>
<td>More focus on design reviews, designs as specs, in-code API docs</td>
</tr>
</tbody>
</table>
| **Design Docs**                             | • Avoid disagreement later in the review cycle  
• Create a “knowledge base” for new developers                                             |
| **API Docs**                                |                                                                                                                                              |
| **Test Lab**                                | Increased Focus on Quality                                                                                                               |
| **OSSTEST**                                 | Share the cost of testing (Past: everyone tested independently)                                                                           |
| **Release Management 4.6**                 | Slightly shorter release cycle  
Harder freeze dates  
Branch master earlier ➔ longer active development period                                |
| **Release Management 4.7**                 | Short and fixed release cycle (June and December)  
Even harder freeze dates: no feature freeze exceptions  
• Make it easier for consumers of Xen to plan their products  
• Decrease the impact of features not making it into Xen x.y |
## Changes proposed/under discussion

<table>
<thead>
<tr>
<th>Change</th>
<th>Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Feature Maturity Lifecycle</strong></td>
<td>• Better understanding of feature maturity for users</td>
</tr>
<tr>
<td></td>
<td>• Encourage more testing: only tested features can be “supported”</td>
</tr>
<tr>
<td></td>
<td>• Find a way to classify non-core features</td>
</tr>
<tr>
<td><strong>Decision Making</strong></td>
<td>• Not optimized for “process and convention changes”</td>
</tr>
<tr>
<td></td>
<td>• Make the process clearer and streamline it</td>
</tr>
<tr>
<td><strong>Review Process Review Criteria</strong></td>
<td>• Contributing to Xen has become harder</td>
</tr>
<tr>
<td></td>
<td>• This just happened, without being discussed, and came as a surprise</td>
</tr>
<tr>
<td></td>
<td>• Caused issues because of mismatching expectations</td>
</tr>
<tr>
<td><strong>Contribution Reporting</strong></td>
<td>• Find better ways to high-light non-code contributions</td>
</tr>
<tr>
<td></td>
<td>• Encourage more code reviews and tests</td>
</tr>
<tr>
<td><strong>Roles / Project Leadership</strong></td>
<td>Conducted a survey in Q3’15: still early days</td>
</tr>
<tr>
<td></td>
<td>• Highlighted different expectations by different people</td>
</tr>
<tr>
<td></td>
<td>• Have a range of options to improve things</td>
</tr>
</tbody>
</table>
Lesson

The project is adapting to a changing environment

Don’t get caught out by changes

Participate in discussions
We are facing new tensions, that require to make conscious trade-offs
New Features
Community Growth
Review Capacity
Review Criteria

Goal: Better Quality & Security
Contributor – Maintainer Interaction
Patches and Comments posted

**Evolution of patch series per month**

**Evolution of comments per month**
Comments per patch / Reviewers

Evolution of the mean (blue) and median (green) comments per patch

Time evolution

Number of patch series

People commenting
We have a problem: more contributions, tougher contribution requirements, same number of reviewers, number of patches under review is growing
We managed to part-fix this through training of new contributors, process changes, better co-ordination.
Tougher requirements on Quality gradually happened

There was no discussion about the quality-contribution trade-off, which led to surprises and some contributors having wrong expectations.

In fact: we didn’t know this was happening until recently.
Implications for Contributors

For new contributors contributing up to smaller 10-15 patches per year:
  – None

For new contributors planning to contributing complex and 15+ patches per year:
  – Reviewers are less willing to review patches without getting something in return

At a minimum:
  – Engage with the Roadmap Process: Communicate your priorities
  – Submit early in the review process and submit designs early for complex code
  – Have realistic expectations

Ideally:
  – Observe patch reviews on xen-devel@ and help with patch reviews of other people’s code
  – Help with testing (test days, test reports, test code)
  – Long term: work towards maintainership of components/features you care about
100 - 500 patches under review at any given time

Larger patches need ACKs from 3-5 people
Cooperation: The paint-gun problem

100 - 500 patches under active review

Patch series A

Patch series B

Patch series N

Reviewers review according to their own schedule and own priorities.

There is no centralized priority list.

You may need to ping reviewers: overdoing this is counter-productive (may be considered as hassling).
New Features
Community Growth

Quality, Security
Different use-cases

Conflicting Requirements

e.g.
cloud / enterprise vs. security vendor
cloud / enterprise vs. embedded vendor
Xen shows off 35-piece cloudpocalypse collection
The latest fixing fashions for open-source hypervisors hit the catwalk

Guest-host escape bug sees Xen project urge rapid upgrade
Xen hypervisor v.4.5.1 offers over 100 fixes and improvements

Xen reports new guest-host escape, this time through CD-ROMs
Don't stick your head in the sand, patch QEMU

Xen warns of new Venom-like vulnerability

Heap overflow in QEMU PCNET NIC lets guests take over hosts.
A newly discovered flaw in the popular open source Xen virtualisation hypervisor layer has prompted the Xen Project to urge users to update from the vulnerable version 4.5 to version 4.5.1 as soon as possible.

According to the Xen Project security team, the XSA-135 flaw is a heap overflow in the Quick Emulator code for the PCNET network interface

Allows admins with limited privileges to take full control
Open source hypervisor developer Xenproject has issued a patch management tool vulnerability that could allow administrators with limited privileges to take full control of the whole host.
Security Scrutiny

Media coverage is just a side-effect.
We care about …
  – There are people out there trying to break Xen
  – And use exploits against Xen users

This means …
  – Code is reviewed with security in mind
  – Think about security when designing a feature
  – Think about security before submitting a patch
  – You may be asked to modify related code that is related to your patch
    (often reviewers code “surrounding” your patch)
Easy Ways to get Involved

Fix some Coverity Scan Issues

- You can get access: see xenproject.org/help/contribution-guidelines.html
- Small, bite-size issues to practice contributing to Xen
Feature Maturity Lifecycle (FML) and Documentation

Proposal @
# FML Requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Implemented</th>
<th>Maintained</th>
<th>Tested</th>
<th>Stable</th>
<th>Documented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preview</td>
<td>Part</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Experimental</td>
<td>Core</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complete (New)</td>
<td>Full</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Supported (New)</td>
<td>Full</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Supported-Legacy-Stable</td>
<td>Full</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
## FML Effects

<table>
<thead>
<tr>
<th></th>
<th>Bugs</th>
<th>Critical bugs block release</th>
<th>Security</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preview</td>
<td>Dev*</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Experimental</td>
<td>Dev*</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Complete (New)</strong></td>
<td>Dev*</td>
<td>No**</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Supported (New)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Supported-Legacy-Stable</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This is a state which has not existed in the past. It is aimed at larger new features, which may only be in use or of interest to a small number of contributors, or where not enough expertise exists in the community to treat the feature as Supported.

*) At developer(s) discretion
**) At Release Managers discretion
FML Goals

**Complete** is aimed at non-core use-cases
- Defuse tensions for non-core features
- Cover for the case where we lose the capability to support

**Supported** requires **automated** testing or **manual** testing during RC phase (otherwise it may be downgraded to Complete)

**Supported-Legacy-Stable** accounts for the fact that many features that existed for a long time, may not be documented or automatically tested
- Phase out over time
FML Status (Nov 26, 2015)

Too many similar states
  – Need to simplify

Some Open Questions
  – Templates and Exact Format of Feature Status
  – Location of files
  – How to handle legacy
Treating Designs Reviews like Code Reviews

Traditionally we treated designs review different to code reviews

- Using PDFs and Text Designs on xen-devel@
- **Issues:** Agreements and changes are not tracked

**Emerging Alternative**

- Post designs as patches in xen.git @ docs/… folders
- Example: xen.git @ docs/misc/xsplice.markup with discussion at lists.xen.org/archives/html/xen-devel/2015-11/msg00244.html
- Using pandoc markdown language and templates
  (see pandoc.org/README.html#pandocs-markdown)
- **Advantages:**
  1. ACKs are tracked ➔ It is clear who agreed with the design
  2. Design evolves with the code ➔ Change the design doc with patches (include into series)
  3. Easy to read and write ➔ Can generate html, pdf’s, etc.