Xen Project Contributor Training
Part 2 : Processes and Conventions

Updated November, 2015

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

lars_kurth

Content

	Principles: Openness, Transparency, Meritocracy
Discussed	Roles: Maintainers, Committers, Project Lead
Updated	Decision Making and Conflict Resolution
Updated	Design reviews
New	Feature Lifecycle and Documentation
Updated	Bug reports and Security Issues
	Patch contribution workflow
	• Anatomy of a good patch series
	• Coding style
Discussed	• Personal repos hosted by Xen Project
Updated	• Staging-to-master pushgate and automated testing
Updated	Release Manager Role and Release Process
	Access to Coverity Scan
Updated	Hackathons, Developer meetings, Ad-hoc meetings to resolve issue
	Earning “status” in the Xen Project community

Principles and Roles:

www.xenproject.org/governance.html

Principles: Openness, Transparency, Meritocracy

Openness: The Xen Project is open to all and provides the same opportunity to all. Everyone participates with the same rules. There are no rules to exclude any potential contributors which include, of course, direct competitors in the marketplace.

Transparency: Project discussions, minutes, deliberations, project plans, plans for new features, and other artifacts are open, public, and easily accessible.

Meritocracy: The Xen Project is a meritocracy. The more you contribute the more respect and responsibility you will earn. Leadership roles in Xen are also merit-based and earned by peer acclaim.

Roles: Maintainers, Committers, Project Lead

- Maintainers
  - Own one or several components in the Xen Project tree
  - Reviews and approves changes that affect their components (Acked-by)
  - It is a maintainer's prime responsibility to review, comment on, co-ordinate and accept patches from other community member's.
  - It is a maintainers prime responsibility to maintain the design cohesion of their components. Which implies: quality, maintainability, system properties, …
- Committer
  - Maintainer with write access to the source tree
  - Acts on the wishes of maintainers (Acked-by)
  - Allowed to act as referees should disagreements amongst maintainers arise
- Project Lead
  - Public face of project
  - Allowed to act as referee should disagreements amongst committers arise

Discuss

Ongoing Discussions

Developer Survey : Improving Xen Project Governance and Conventions

- Part 1 :
  Hierarchy of maintainers in the xen.git MAINTAINERs file
- Part 2 :
  Trust amongst different stakeholders in the peer review process
- Part 3 :
  Other related Governance Issues (Voting Model, Decision Making)
- Results of Phase 1 of the Review Process study

Decision Making:

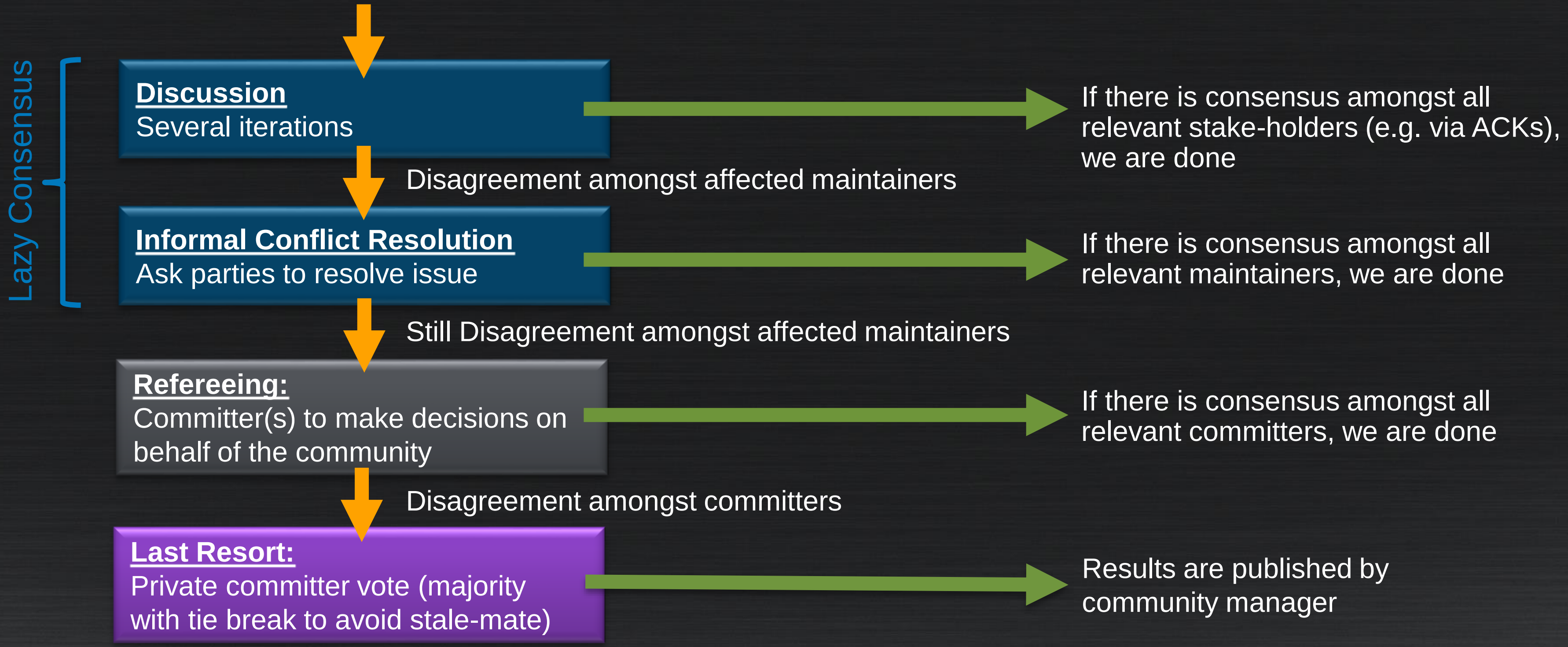
www.xenproject.org/governance.html

Update

New

Informal Workflow for Code

Applies to: Design Discussions, Code Reviews, Documentation, …



Lazy Consensus

Discussion
Several iterations

If there is consensus amongst all relevant stake-holders (e.g. via ACKs), we are done

Disagreement amongst affected maintainers

Informal Conflict Resolution
Ask parties to resolve issue

If there is consensus amongst all relevant maintainers, we are done

Still Disagreement amongst affected maintainers

Refereeing:
Committer(s) to make decisions on behalf of the community

If there is consensus amongst all relevant committers, we are done

Disagreement amongst committers

Last Resort:
Private committer vote (majority with tie break to avoid stale-mate)

Results are published by community manager

Decision Making: Lazy Consensus

Lazy consensus is a mutual consent decision making process between you and the community that states your default support for a proposal is "yes", unless you explicitly say "no".

Restrictions:

- Any "No" statement must be accompanied by an explanation
- Excludes decisions that require a sign-off, e.g. an Acked-by a maintainer on a patch. But, if there are several maintainers that need to agree, the affected maintainers operate using Lazy Consensus
- Excludes formal voting, e.g. committer election, governance changes, …

Assumptions:

- Require everyone who cares for the health of the project to watch what is happening, as it is happening.

Lazy Consensus: Examples

The design discussion leading to this point in time was lengthy with some disagreements amongst core developers. It took 2 months and 43 email exchanges to get to the point below.

The example shows how the proposer prompting the Maintainers for clarification on whether all the issues have been resolved and by doing so, got to an agreement.

```
[Contributor]
In this case, if libvirt/XenAPI is trying to query a domain's cache utilization in the system
(say 2 sockets), then it will trigger _two_ such MSR access hypercalls for CPUs in the 2 different
sockets.
```

`If you are okay with this idea, I am going to implement it.`

```
[Maintainer]
I am okay with it, but give it a couple of days before you start so that others can voice their
opinions too. Dom0 may not have a vcpu which is scheduled/schedulable on every socket.

[snip: ... there was another short exchange clarifying a question, which were addressed during the
conversation]

[Contributor]
No more comments on this MSR access hypercall design now, so I assume people are mostly okay with it?

[Another Maintainer]
Yes -- I think everyone who commented before is satisfied with that approach, and anyone who hasn't
commented has had ample opportunity to do so
```

Conflict Resolution : Informal

Maintainers and core developers sometimes hold different opinions regarding architecture, design or other issues. That is entirely normal. Because such situations can delay progress and turn away contributors, the project has some mechanisms to resolve this.

Informal:

- Ask the parties that disagree to resolve the disagreement
- Leave some “reasonable” time period to allow the disagreeing parties to come to a conclusion
- If in doubt, you can ask one of the committers and/or the community manager for advice.

This works in most cases.

Conflict Resolution : Formal

In situations when no resolution can be found informally, refereeing can be used. People with higher “status” in the community can make decisions on behalf of the community.

Example: Two maintainers disagree on a design. You have asked for the issue to be resolved, but no resolution could be found.

Formal:

- Ask the referee: in this case the committer(s) responsible for the maintainers to resolve the disagreement
- If committers can’t agree on a way forward, a private formal majority vote amongst committers can be used to break the dead-lock
- If in doubt, ask the community manager for advice.

Factors to Consider

- Referees do not always proactively step in and resolve an issue
  - Workload : may have missed a disagreement
  - Resolving issues is harder for some people than others
- Asking for an issue resolution in public or private?
  - Prompting disagreeing parties to resolve an issue publicly is not always easy
  - It is OK, to ask a referee or the community manager for advice privately
  - However, Transparency and Lazy Consensus require that discussions and decisions are made in public. This means that
    - It is OK to do preparation work to resolve a conflict in private (e.g. on IRC, a phone call, etc.)
    - But, in such a cases, there needs to be a clear statement on the list that shows the outcome and invites others to provide feedback

Example

A maintainer stepped in to understand and resolve an issue by having a quick conversation with one party involved: he stated the fact that there was a private discussion, summarized it and invited others to comment.

```
[Maintainer]
So XXX and I had a chat about this [the disagreement that came up
in a discussion], and I think we came up with something that would
be do-able. (This is from memory, so XXX please correct me if I
missed anything).

So the situation, as I understand it, is:

...

That should be a good balance -- it's not quite as good as having
as separate daemon, but it's a pretty good compromise.

Thoughts?
```

New

Informal Workflow for Other Purposes

Applies to: Proposals related to Development Practices and Processes

Consensus Approval

Informal Discussion of a Proposal
Possibly several options with an informal vote on each option to establish momentum for each option

If there is clear consensus amongst all relevant stake-holders we are done

No clear agreement for a single option
Next step: Proposer makes a formal proposal with one option

Informal or Formal Committer Vote
A single proposal marked [Vote] on a single concrete proposal

If there is consensus amongst relevant committers, we are done

No agreement for a single option
Next step: Proposer asks community manager for resolution

Last Resort:
Private committer vote (majority with tie break to avoid stale-mate)

Results are published by community manager

New

Current Voting Process

Voting is done with numbers:

```
+1 : a positive vote
 0 : abstain, have no opinion
-1 : a negative vote
```

A negative vote should include an alternative proposal or a detailed explanation of the reasons for the negative vote.

Issues:

A single -1 is essentially a veto and will block any vote, except for a “last resort” vote. In practice, committers wanted to record that they are against a proposal without blocking it

Discuss

Improved (minimal) Process Proposal

Voting is done with numbers:

```
+2 : agree, but care strongly enough to argue for it
+1 : agree, but don't care enough to argue for it
 0 : abstain, have no opinion
-1 : disagree, but don't care enough to argue against it
-2 : disagree, but feel strongly enough to argue against it
```

Only a -2 will stop an agreement.

Note: Although this proposal has wide agreement (see lists.xenproject.org/archives/html/xen-devel/2015-10/msg01885.html), a formal vote is required and some details need to be resolved, as we are thinking to move to a majority based model.

New

Formal Votes

Applies to:

- Committer, Project Lead and Release Manager Elections
- Governance changes (for governance published on xenproject.org)

Issues:
Recently, we have conducted a survey, which highlighted that the decision making portions of the governance are not very clear and there is space to streamline the process. We are working on a proposal to clarify and streamline decision making.

Updated

Design Reviews:

Undocumented Convention

Design Reviews

The Project has no formal requirement to submit designs before a patch

BUT:

- Designs are welcome, for complex designs
- Sometimes a fully fledged design is not necessary : a set of questions, can iteratively lead a design
- If in doubt, as to whether a design is necessary ask the community for input

Example: Question leading to a design

See “cpufreq implementation for OMAP under xen hypervisor”

```
[Contributor]
Hi to all.

I want to implement an cpufreq support for OMAP processors in xen. I use the
Linux kernel as Dom0.

I know that there are 2 implementations of cpufreq: Domain0 based cpufreq and
Hypervisor based cpufreq. But those implementations are made only for x86
architecture, not for the ARM architecture.

Could anybody give me an advice how to do that?
```

After an initial answer, the proposal was iteratively improved leading to a design.

The design turned out to be more complex than anticipated because of dependencies with Linux and architectural differences between x86 and ARM.

Example: Fully Fledged Design

See "FIFO-based event channel ABI design (draft B)"

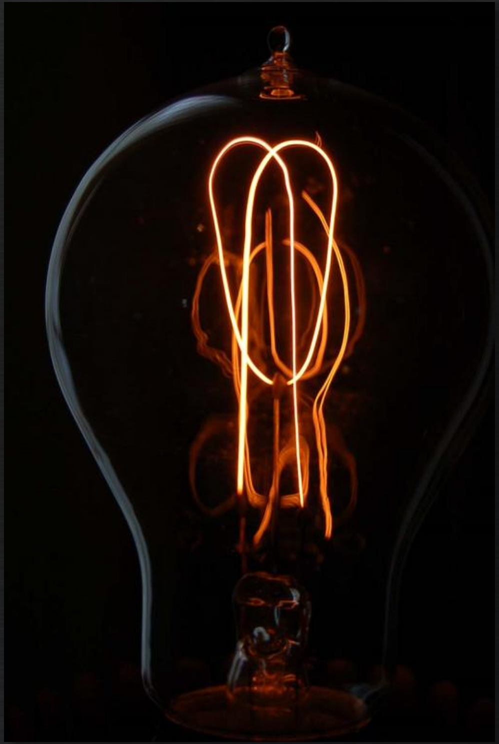
- This was an example of a fully fledged detailed design.
- Note that there was a version A beforehand – a precursor of draft B
- The design was competing with an entirely different design by Wei Liu, for which code already existed – some of which had been posted and reviewed

The community had to make a decision, which design to go for.

- It turned out that a prototype could be put together relatively quickly and an RFC followed.
- This then led to a community decision to go for the FIFO-based event channel (with the agreement of Wei Liu)

If you compare the amount of questions, these were similar to the previous example, but took considerably less elapsed time to review.

This was mainly due to the fact that most developers were within one time-zone and that the design was well thought through.


Vinovyn @ Flickr

Considerations:

There is no right or wrong approach

A more iterative design review (based on a problem or idea which is not fully defined),

- may be less predictable and
- depends on the quality of communication between proposer and reviewers

A fully fledged design,

- may require significant re-work if there are issues with the use-cases, wrong assumptions, etc.

Design as Documentation

Updated

Example: FIFO-based event channel ABI design

- The FIFO-based event channel ABI design had a further 6 revisions (up to draft H)
- It was kept up-to-date with the implementation. See
  - lists.xenproject.org/archives/html/xen-devel/2013-11/msg01414.html referring to
  - xenbits.xen.org/people/dvrabel/event-channels-H.pdf
- The design doc now serves as detailed documentation for the ABI.
- Recommended Location for Feature Specs and Docs
  - xen.git @ docs/specs
  - xen.git @ docs/features

New

Feature Lifecycle and Documentation

Proposal @http://lists.xenproject.org/archives/html/xen-devel/2015-11/msg00609.html

New

Proposal: Clear Criteria for Features

Clearly establish criteria for: Feature / Platform is

- fully implemented, maintained, tested, stable (APIs) & documented

Based on above criteria, award support status

- Preview, Experimental, Complete (new), Supported (new), Supported-Legacy-Stable (the old Supported) and Deprecated

Which in effect controls

- How bugs and issues are handled
- Whether regressions and blockers block Xen Releases
- Whether security issues would be handled by the security team.

Document Feature Status in [xen.git @ docs/features]()

Bug Reports:

wiki.xenproject.org/wiki/Reporting_Bugs_against_Xen_Project

Guilherme Tavares @ Flickr

E-mail based bug reporting Process

Raise Issue or [BUG]:

Description of issue with supporting information, such as environments, logs, etc. to xen-devel or xen-users

IMPORTANT: suspected security vulnerabilities are reported to security@xenproject.org only

Clarification:

Community may ask some more questions to clarify the issue and determine whether the issue in question is a bug or not.

Maintainer confirms issue as bug that needs to be tracked

More Information:

Raiser of bug provides more information

Tracked bug:

Maintainer adds bug to bugs.xenproject.org

Fixing:

Community member fixes the bug using the contribution workflow. Once the fix is committed and the bug confirmed fixed the maintainer closes the bug

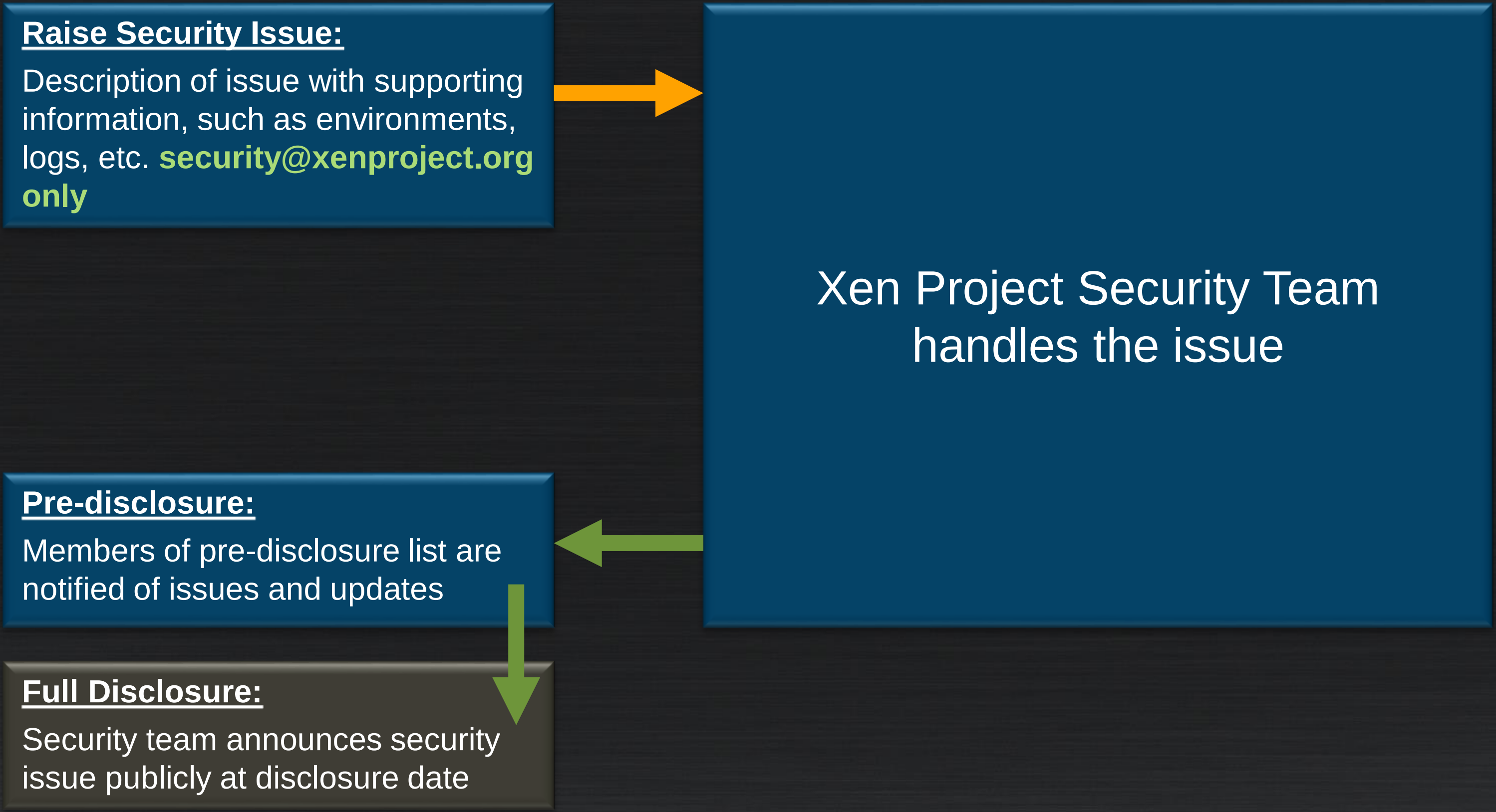
Closed

Security Vulnerabilities:

www.xenproject.org/security-policy.html

Guilherme Tavares @ Flickr

Reporting Security Bugs



Raise Security Issue:

Description of issue with supporting information, such as environments, logs, etc. **security@xenproject.org only**

Xen Project Security Team handles the issue

Pre-disclosure:

Members of pre-disclosure list are notified of issues and updates

Full Disclosure:

Security team announces security issue publicly at disclosure date

Updated

Security Process:

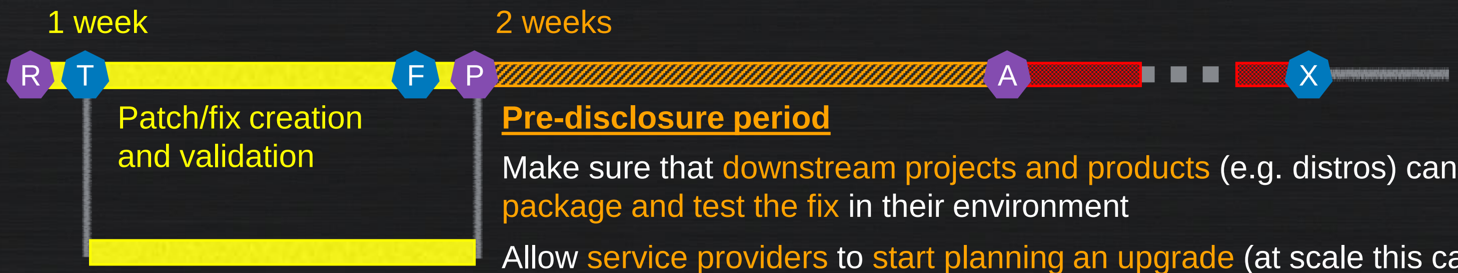
www.xenproject.org/security-policy.html

Credit for this section of the talk: George Dunlap

Guilherme Tavares @ Flickr

Xen Project Responsible Disclosure

Updated



Patch/fix creation and validation

Description, CVE allocation, …

Pre-disclosure period

Make sure that downstream projects and products (e.g. distros) can package and test the fix in their environment

Allow service providers to start planning an upgrade (at scale this can take a week)

Allow service providers to deploy an upgrade before the embargo completes (specified in each advisory whether applicable)

R: Vulnerability Reported
T: Triage
P: Vulnerability Pre-disclosed
A: Vulnerability Announced
F: Fix Available
X: Fix Deployed

🟨 Vulnerability is known by the reporter and the security team
Note: It may also be known and used by black hats

▨ Vulnerability is known about by a privileged and small group of users

🟥 Vulnerability is known publicly

Goals of the Security Process

- Encourage people to report bugs responsibly
- Minimize time that users are vulnerable to attack
- Maintain trust in the community

Updated

Policies

Timing of disclosure

- Honor the wishes of the reporter
- Suggest time period: 1 week to fix, 2 weeks pre-disclosure

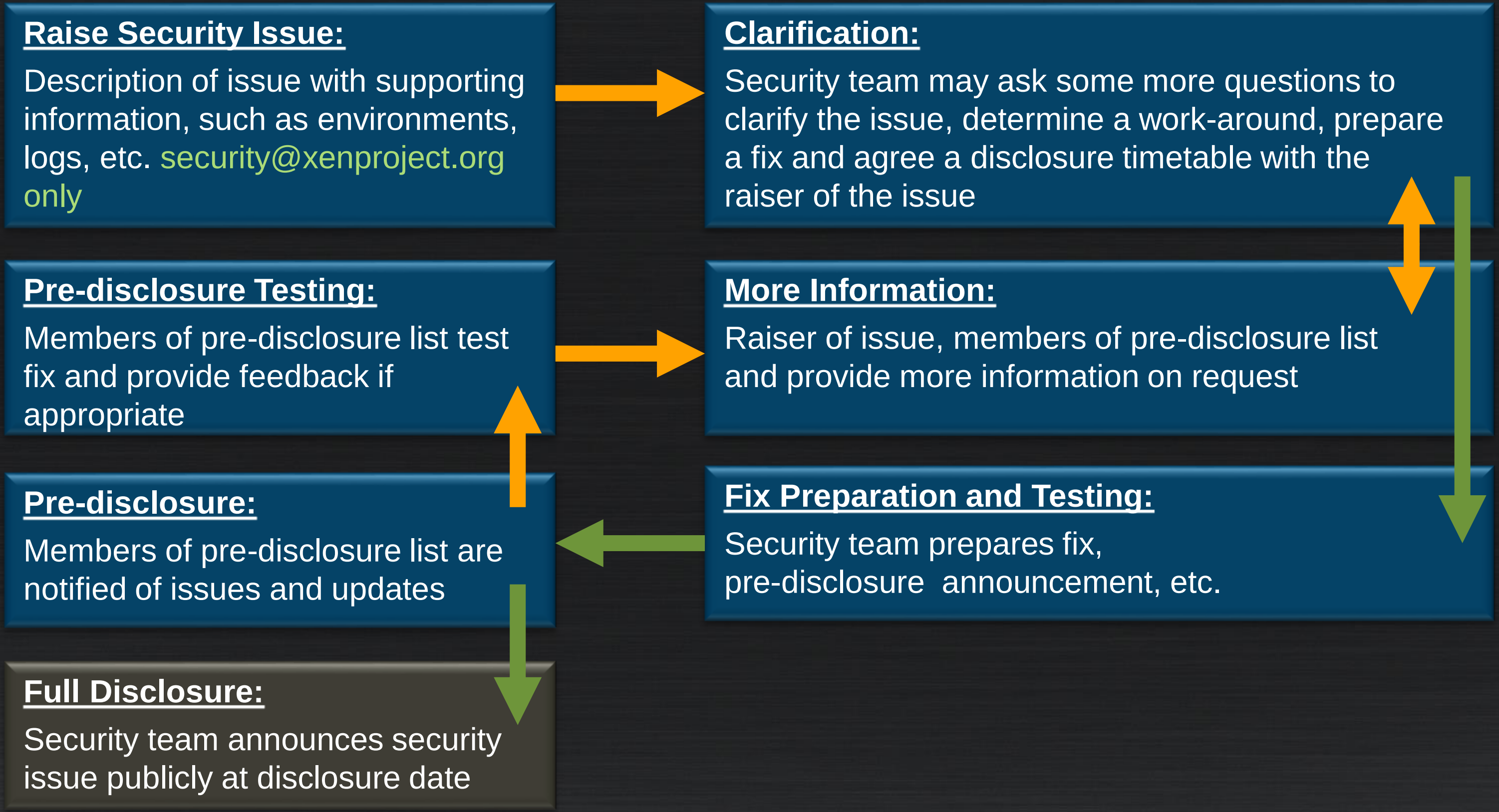
Pre-disclosure list

- Open to any “genuine provider” of software / service using Xen

Xen Project Security Team

- Distinguished Community Members
- Read vulnerability reports
  - Determine if it is a vulnerability
  - Come up with a fix (bringing in others if necessary)
  - Coordinate disclosure
- Manage predisclosure list according to policy

Security Bugs (revisited)



Raise Security Issue:

Description of issue with supporting information, such as environments, logs, etc. security@xenproject.org only

Clarification:

Security team may ask some more questions to clarify the issue, determine a work-around, prepare a fix and agree a disclosure timetable with the raiser of the issue

More Information:

Raiser of issue, members of pre-disclosure list and provide more information on request

Fix Preparation and Testing:

Security team prepares fix, pre-disclosure announcement, etc.

Pre-disclosure:

Members of pre-disclosure list are notified of issues and updates

Pre-disclosure Testing:

Members of pre-disclosure list test fix and provide feedback if appropriate

Full Disclosure:

Security team announces security issue publicly at disclosure date

What happens if the Security Team

- …doesn’t honor the wishes of the reporter?
  - If the reporter doesn’t trust the process, they may go with full disclosure
- …favors some group in the community (aka is not impartial)?
  - Massive loss of trust in the Xen Project
  - Possible legal repercussions for anti-trust violations

New

Additional Resources

Open Source Security Practices on Linux.com

- Part 1: A Cloud Security Introduction
- Part 2: Containers vs. Hypervisors - Protecting Your Attack Surface
- Part 3 and 4 will be published shortly

Also of interest:

- eWeek: How Xen Manages Security
- Are Today's FOSS Security Practices Robust Enough in the Cloud Era?