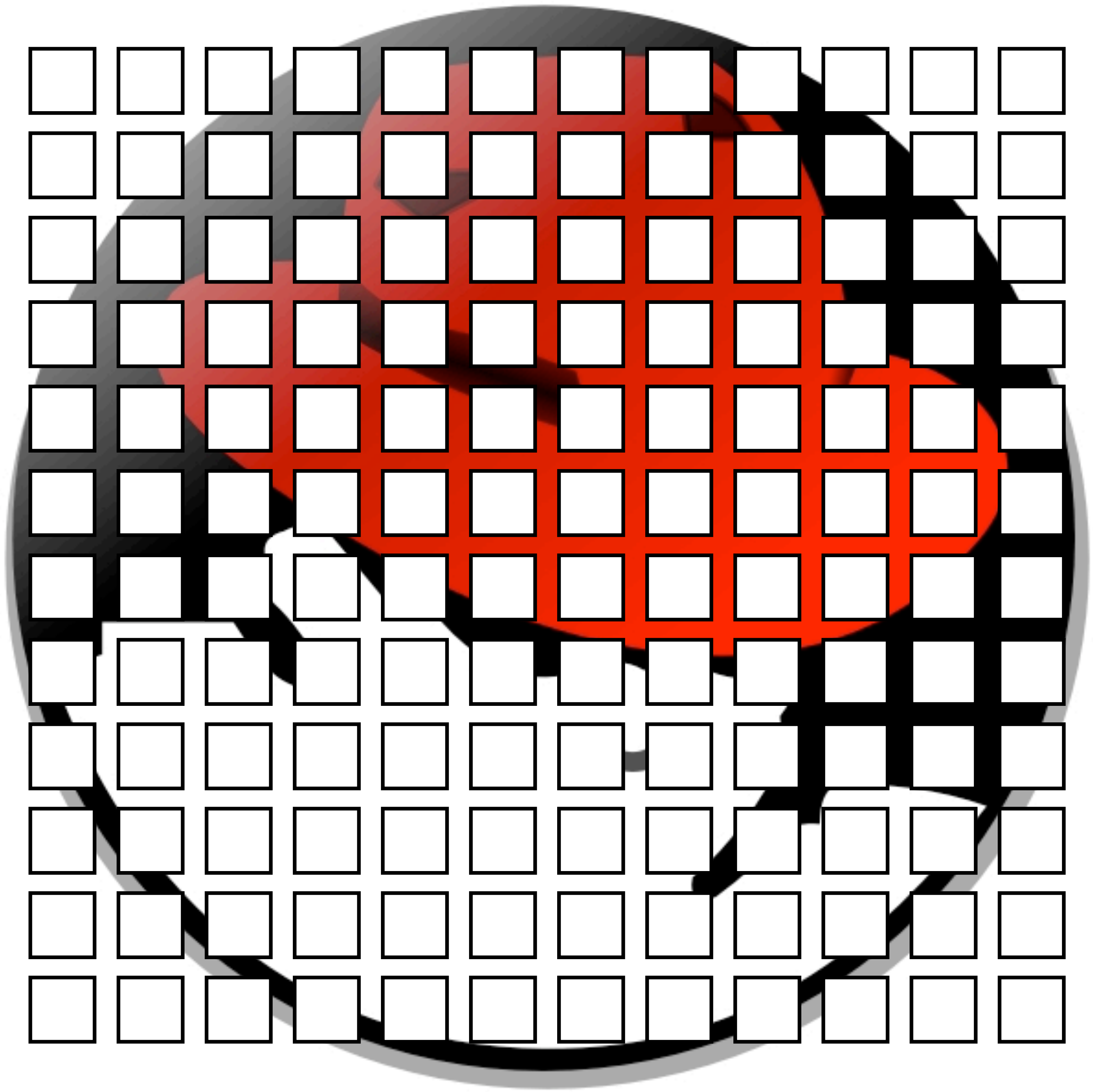
The Beauty and the Beast

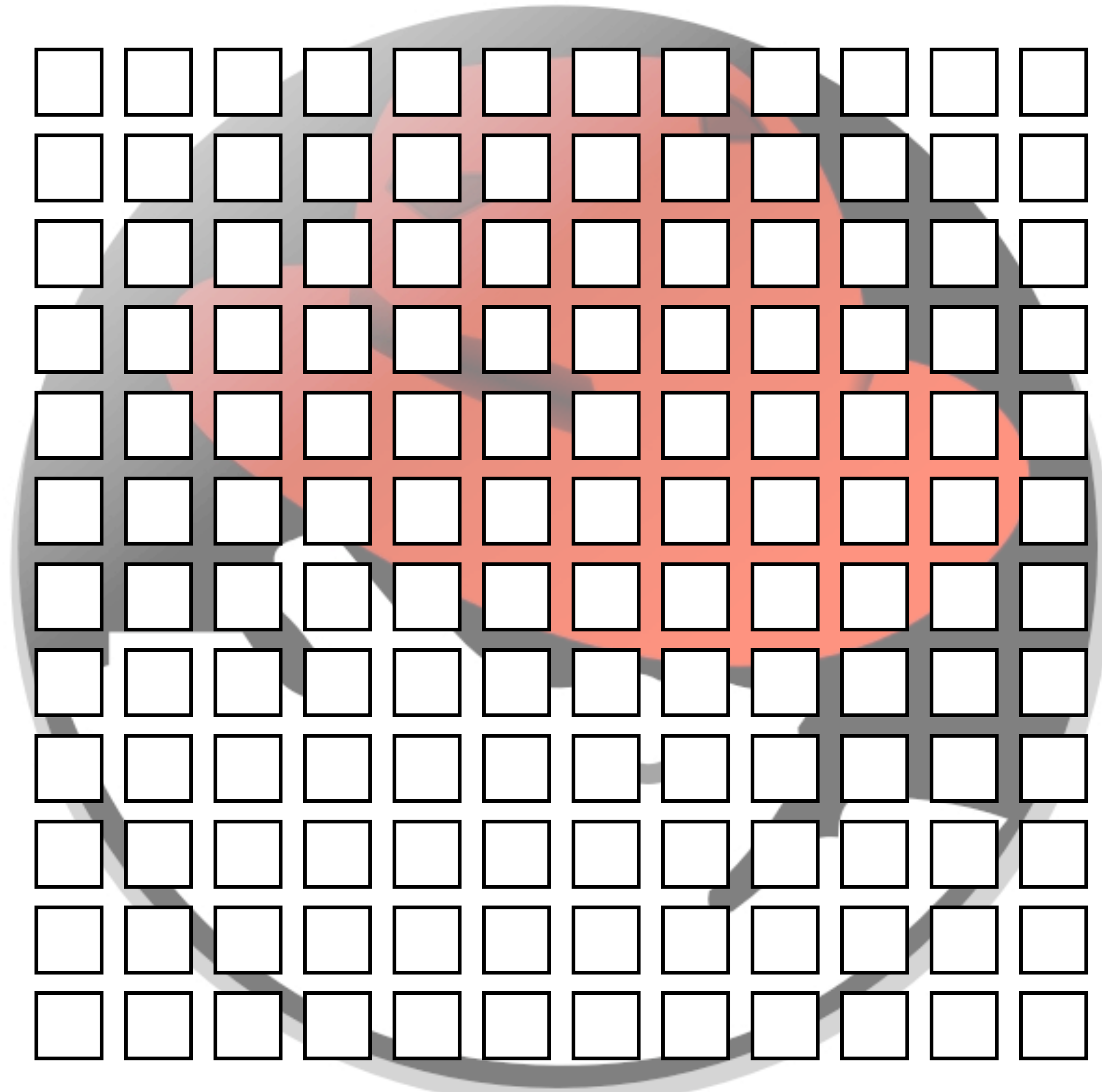
Vulnerabilities in Red Hat's Packages

Stephan Neuhaus <Stephan.Neuhaus@disi.unitn.it>
Thomas Zimmermann <tzimmer@microsoft.com>

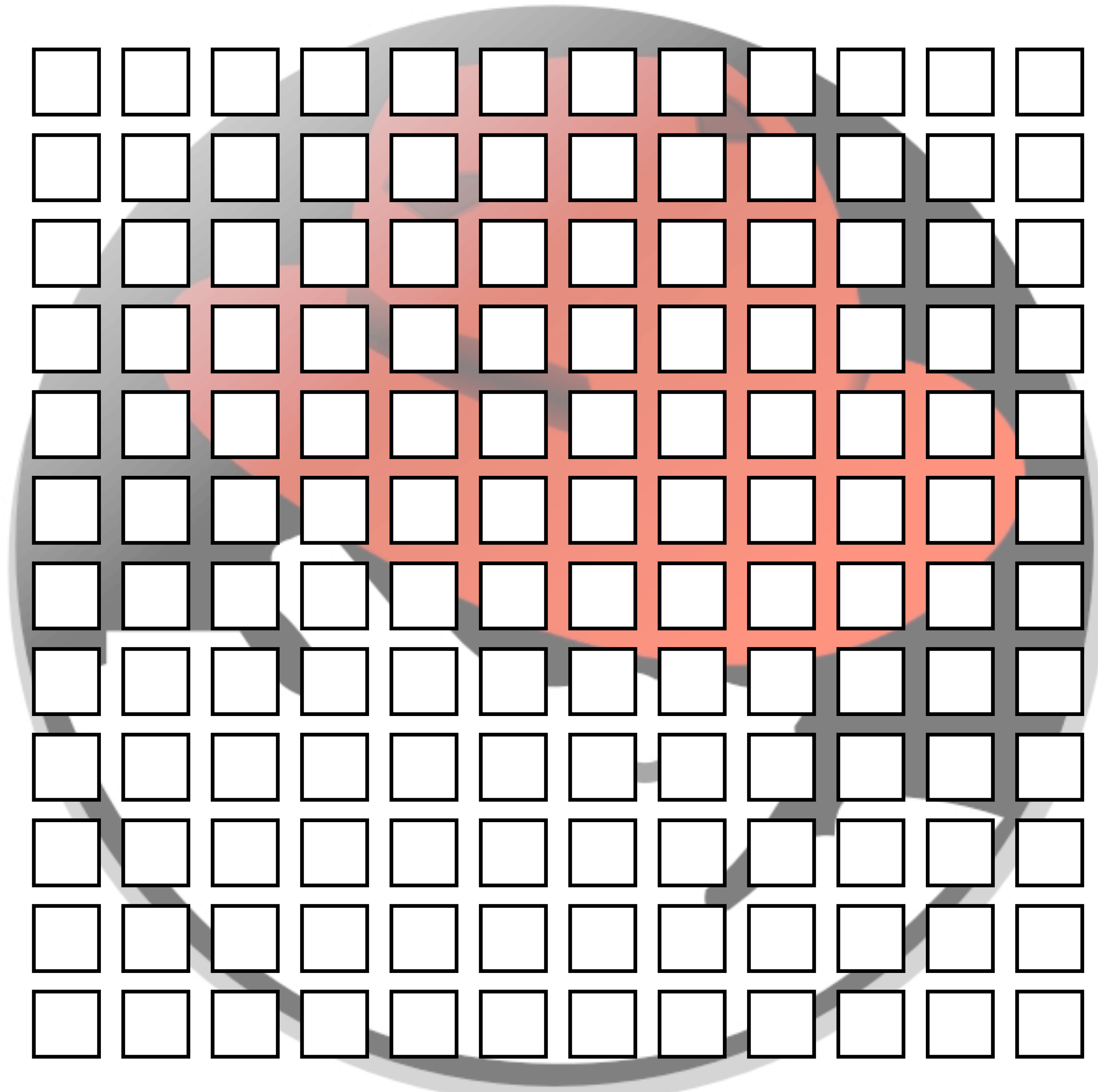
Vulnerabilities are important because fixing them costs a lot of money (2005 FBI study: 67 Bn $). There are 3241 packages (or were, by August 2008) offered by Red Hat. (There are certainly more being offered for Red Hat!)



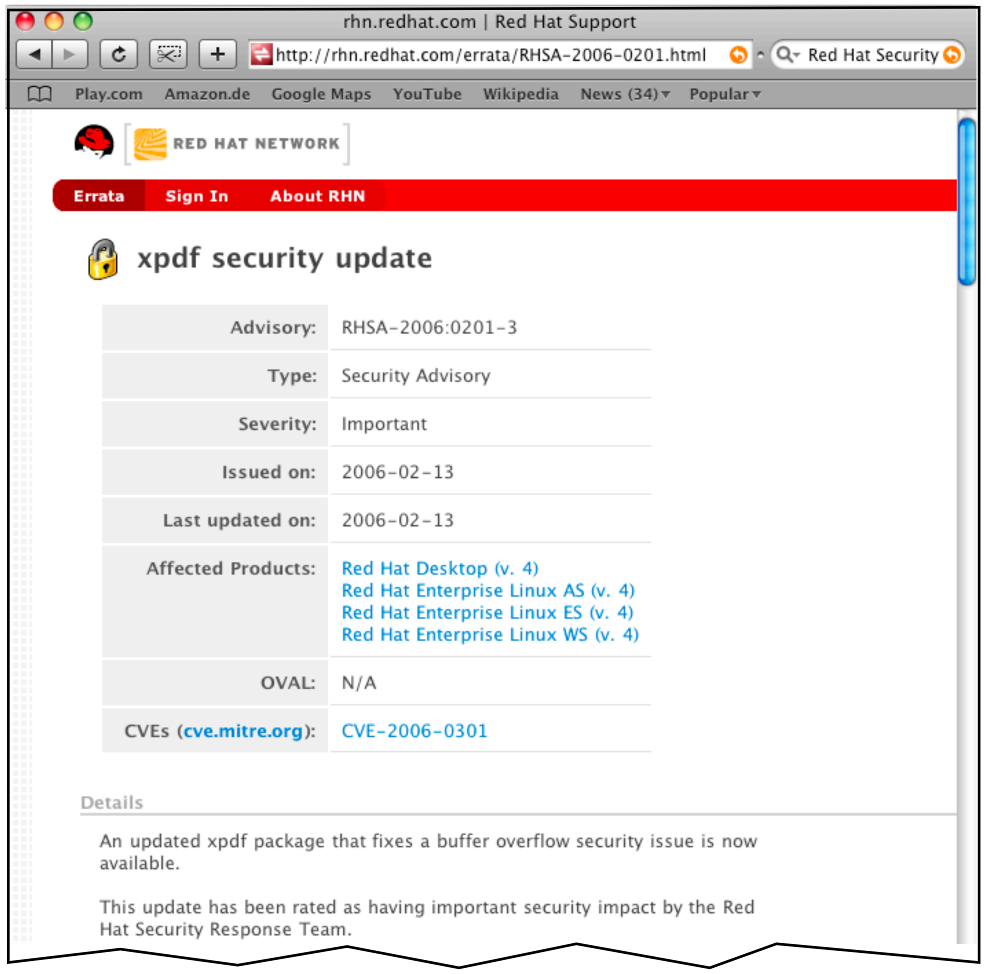
Vulnerabilities are important because fixing them costs a lot of money (2005 FBI study: 67 Bn $). There are 3241 packages (or were, by August 2008) offered by Red Hat. (There are certainly more being offered for Red Hat!)



Vulnerabilities are important because fixing them costs a lot of money (2005 FBI study: 67 Bn $). There are 3241 packages (or were, by August 2008) offered by Red Hat. (There are certainly more being offered for Red Hat!)



Explain colours: white = no vulnerabilities, blue -> red: progressively more


rhn.redhat.com | Red Hat Support

http://rhn.redhat.com/errata/RHSA-2006-0201.html

Red Hat Security

Play.com Amazon.de Google Maps YouTube Wikipedia News (34) Popular

RED HAT NETWORK

Errata Sign In About RHN

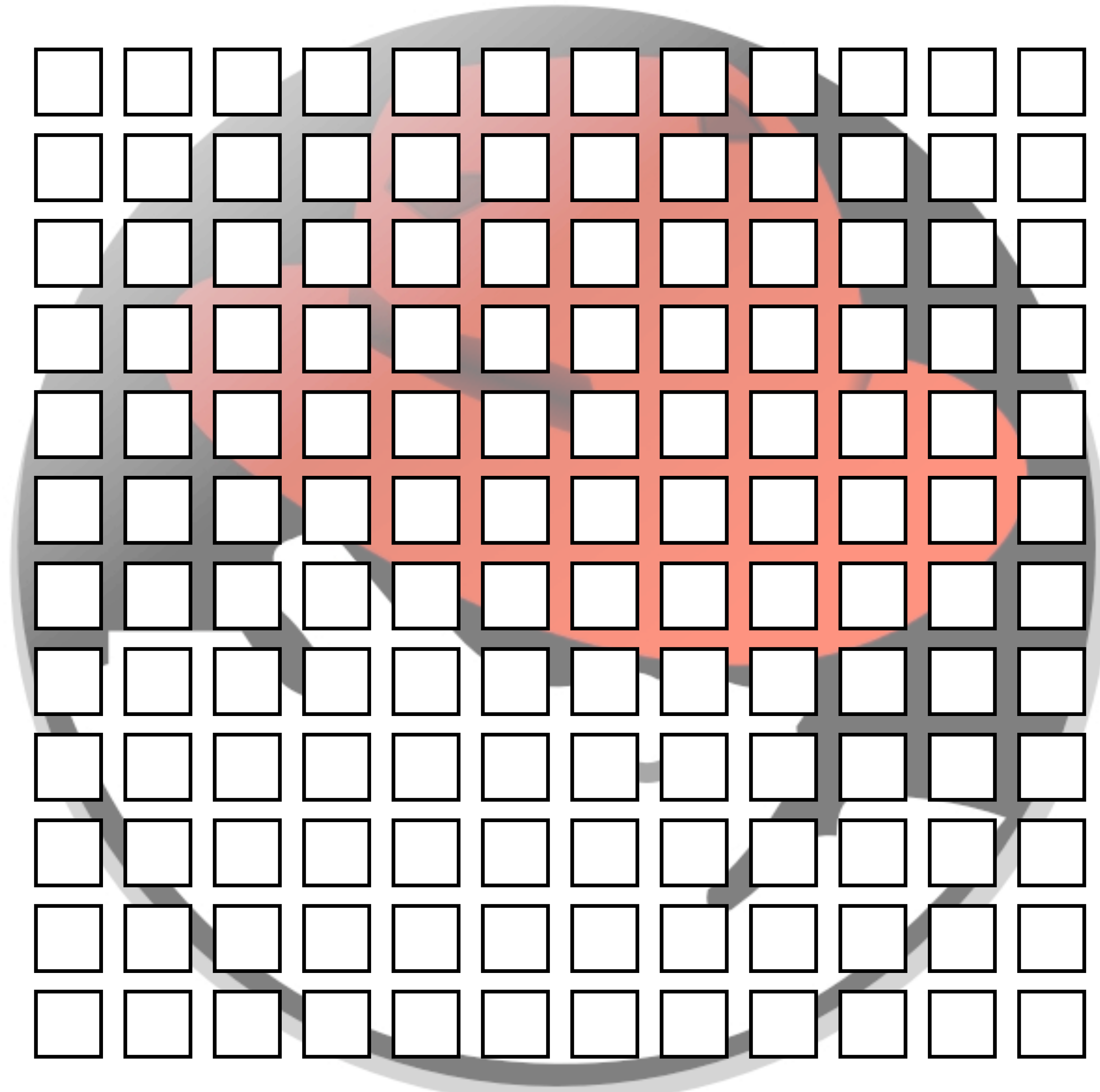
xpdf security update

Advisory:	RHSA-2006:0201-3
Type:	Security Advisory
Severity:	Important
Issued on:	2006-02-13
Last updated on:	2006-02-13
Affected Products:	Red Hat Desktop (v. 4) Red Hat Enterprise Linux AS (v. 4) Red Hat Enterprise Linux ES (v. 4) Red Hat Enterprise Linux WS (v. 4)
OVAL:	N/A
CVEs (cve.mitre.org):	CVE-2006-0301

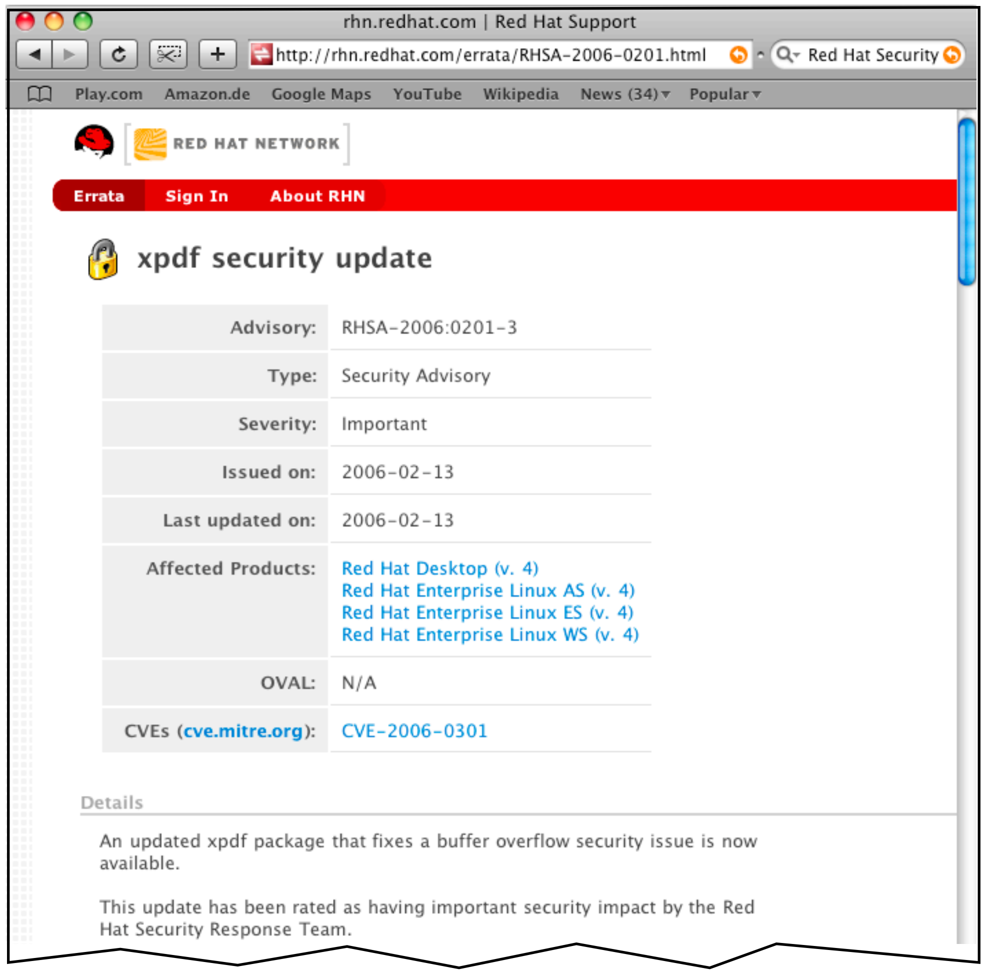
Details

An updated xpdf package that fixes a buffer overflow security issue is now available.

This update has been rated as having important security impact by the Red Hat Security Response Team.



Explain colours: white = no vulnerabilities, blue -> red: progressively more



rhn.redhat.com | Red Hat Support

http://rhn.redhat.com/errata/RHSA-2006-0201.html

Red Hat Security

Play.com Amazon.de Google Maps YouTube Wikipedia News (34) Popular

RED HAT NETWORK

Errata Sign In About RHN

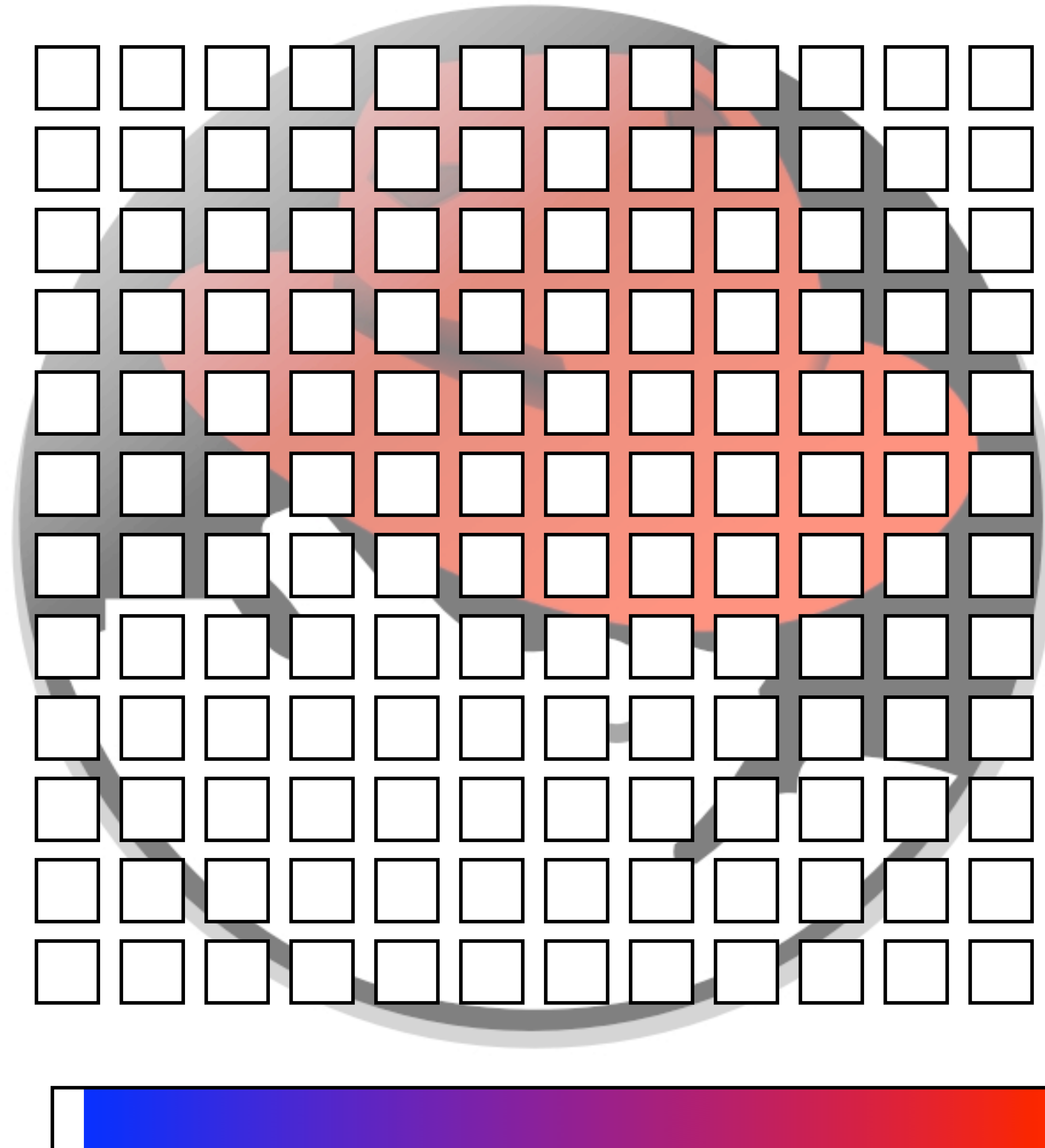
xpdf security update

Advisory:	RHSA-2006:0201-3
Type:	Security Advisory
Severity:	Important
Issued on:	2006-02-13
Last updated on:	2006-02-13
Affected Products:	Red Hat Desktop (v. 4) Red Hat Enterprise Linux AS (v. 4) Red Hat Enterprise Linux ES (v. 4) Red Hat Enterprise Linux WS (v. 4)
OVAL:	N/A
CVEs (cve.mitre.org):	CVE-2006-0301

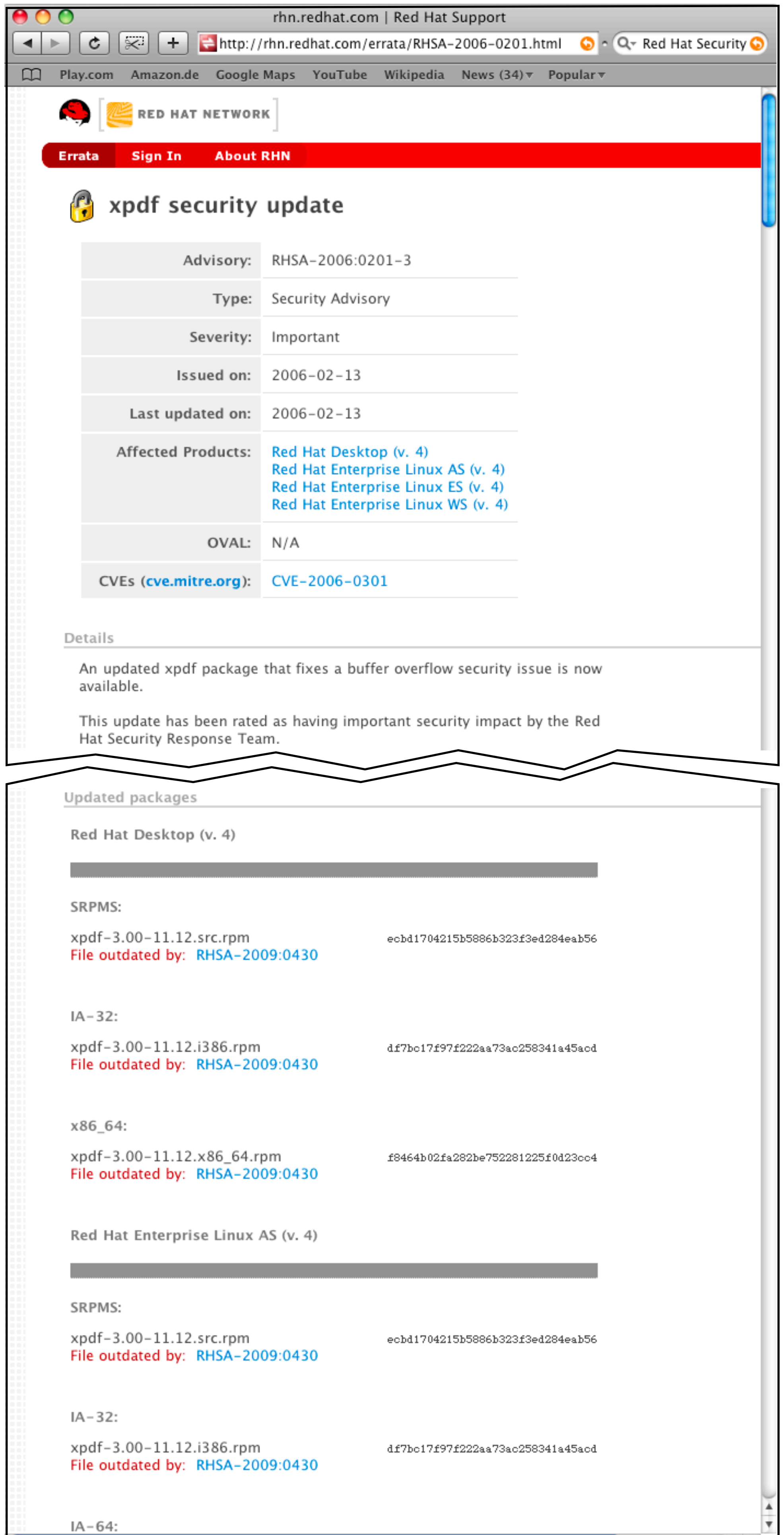
Details

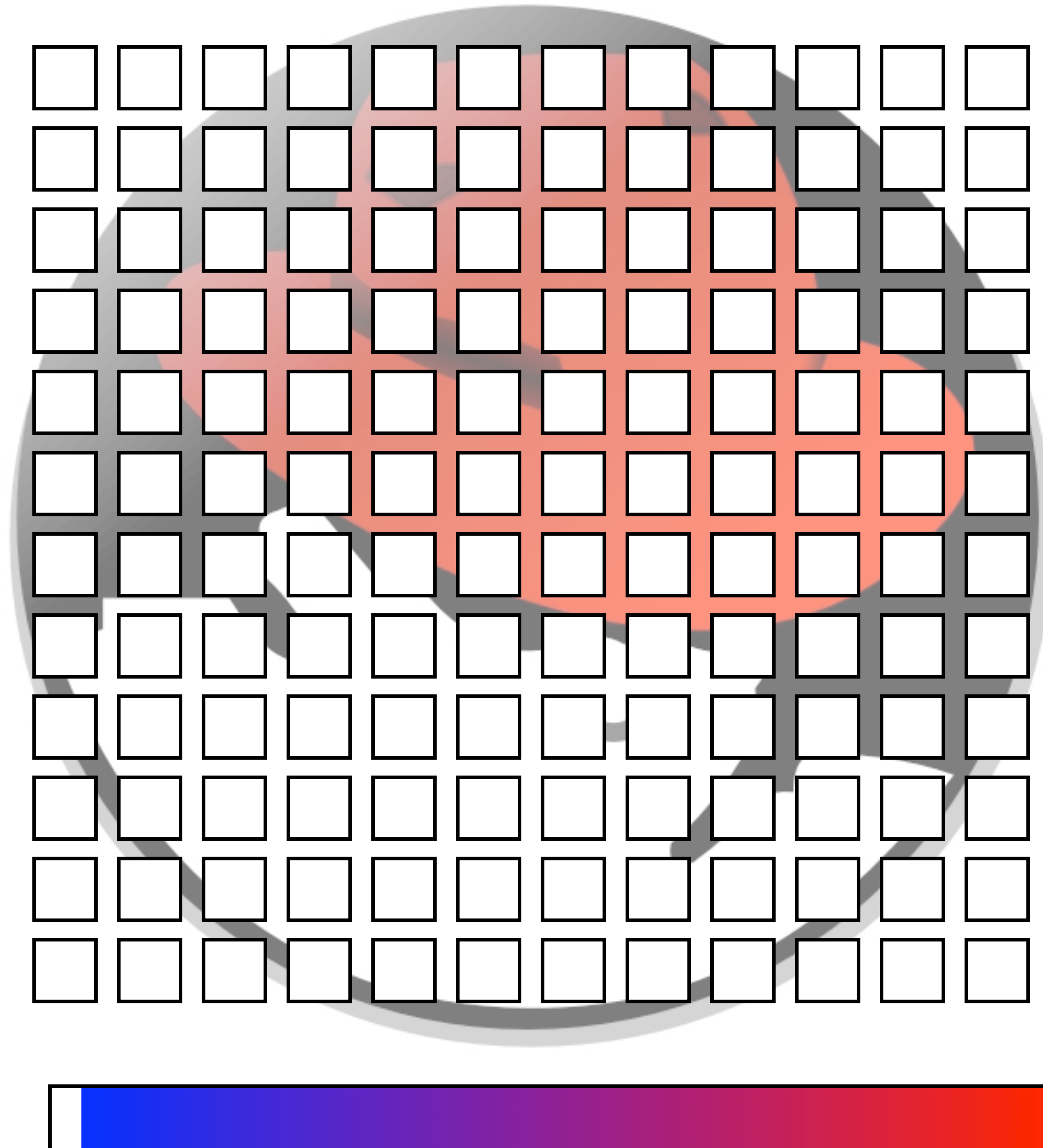
An updated xpdf package that fixes a buffer overflow security issue is now available.

This update has been rated as having important security impact by the Red Hat Security Response Team.

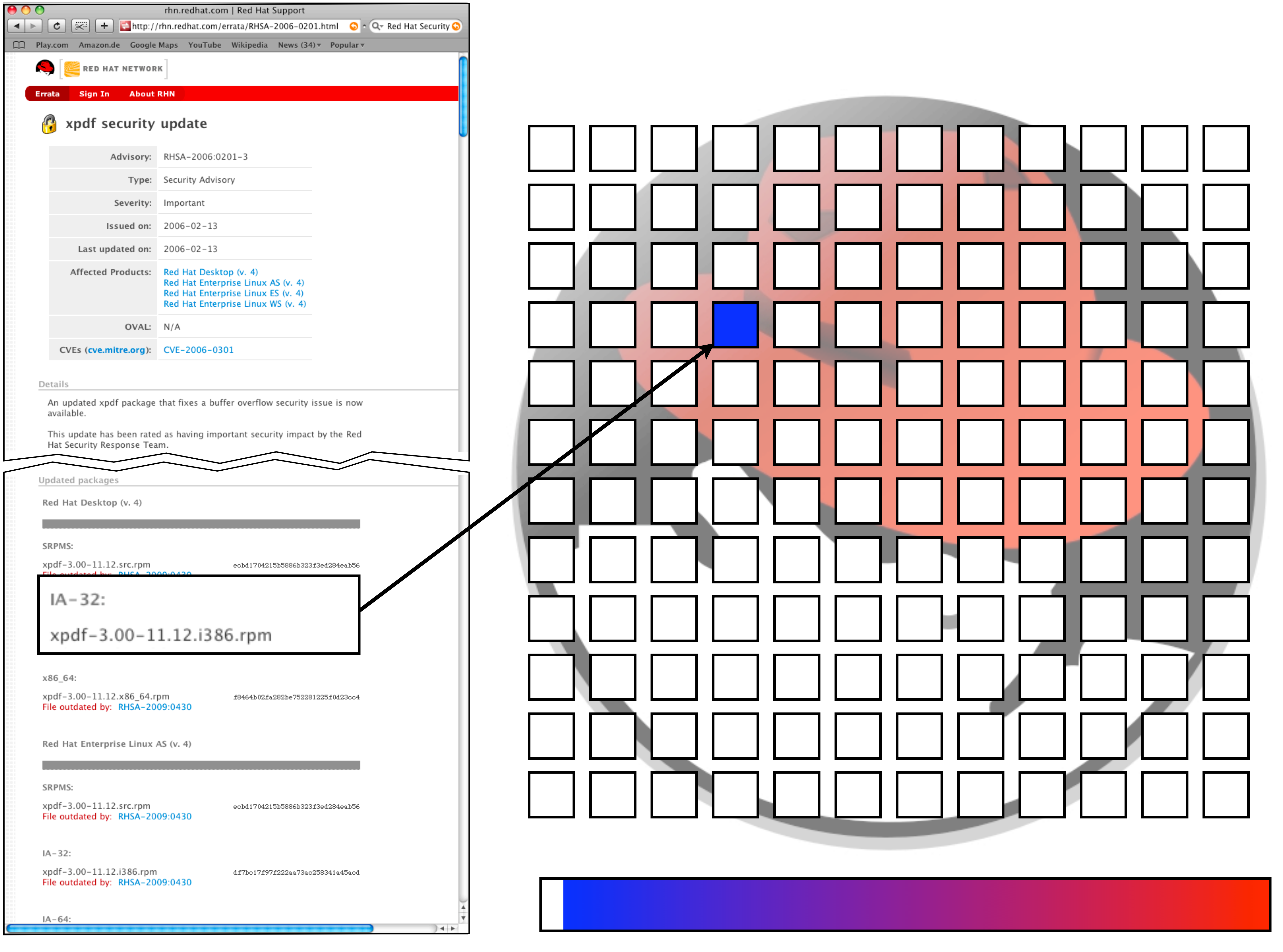


Explain colours: white = no vulnerabilities, blue -> red: progressively more

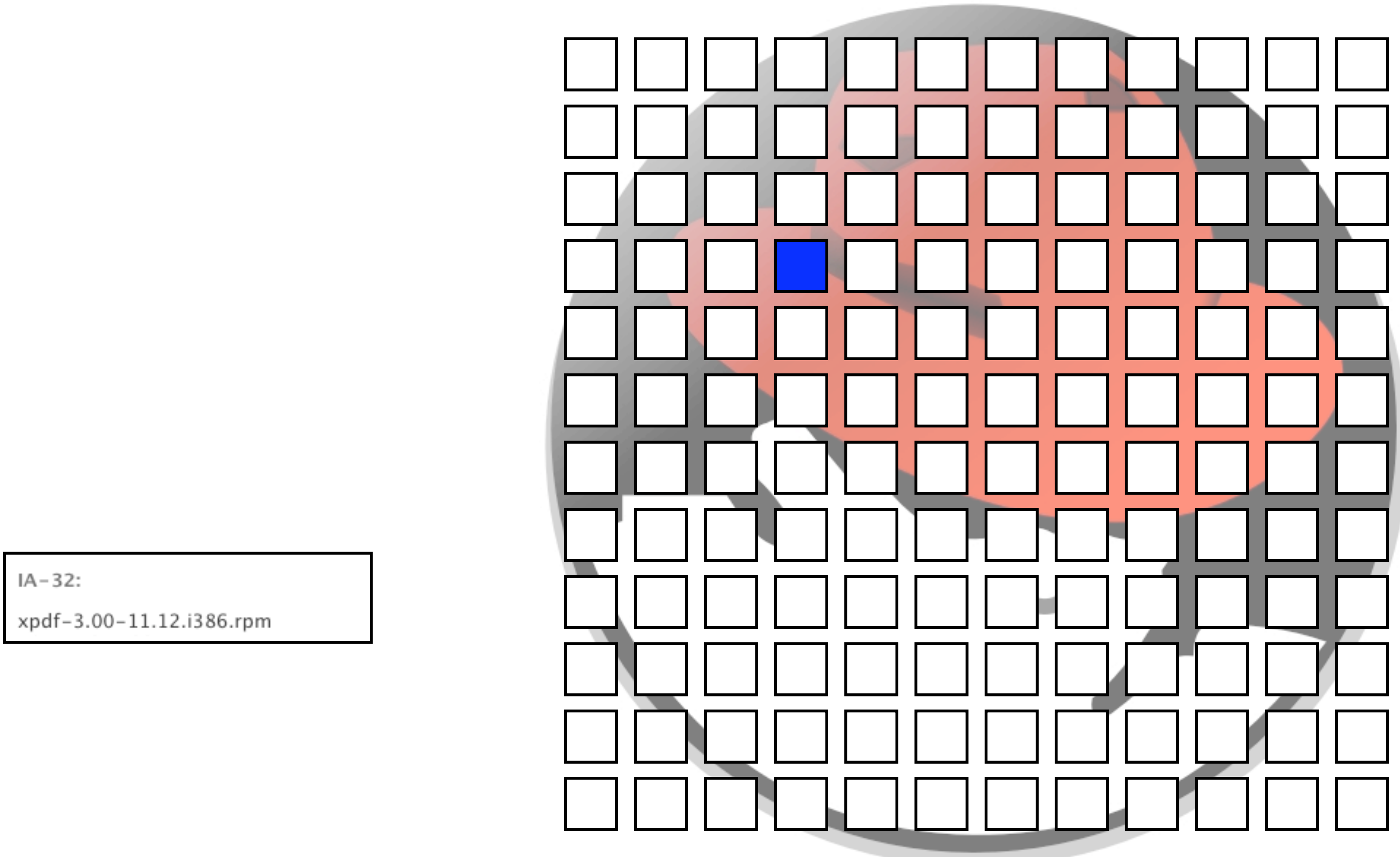




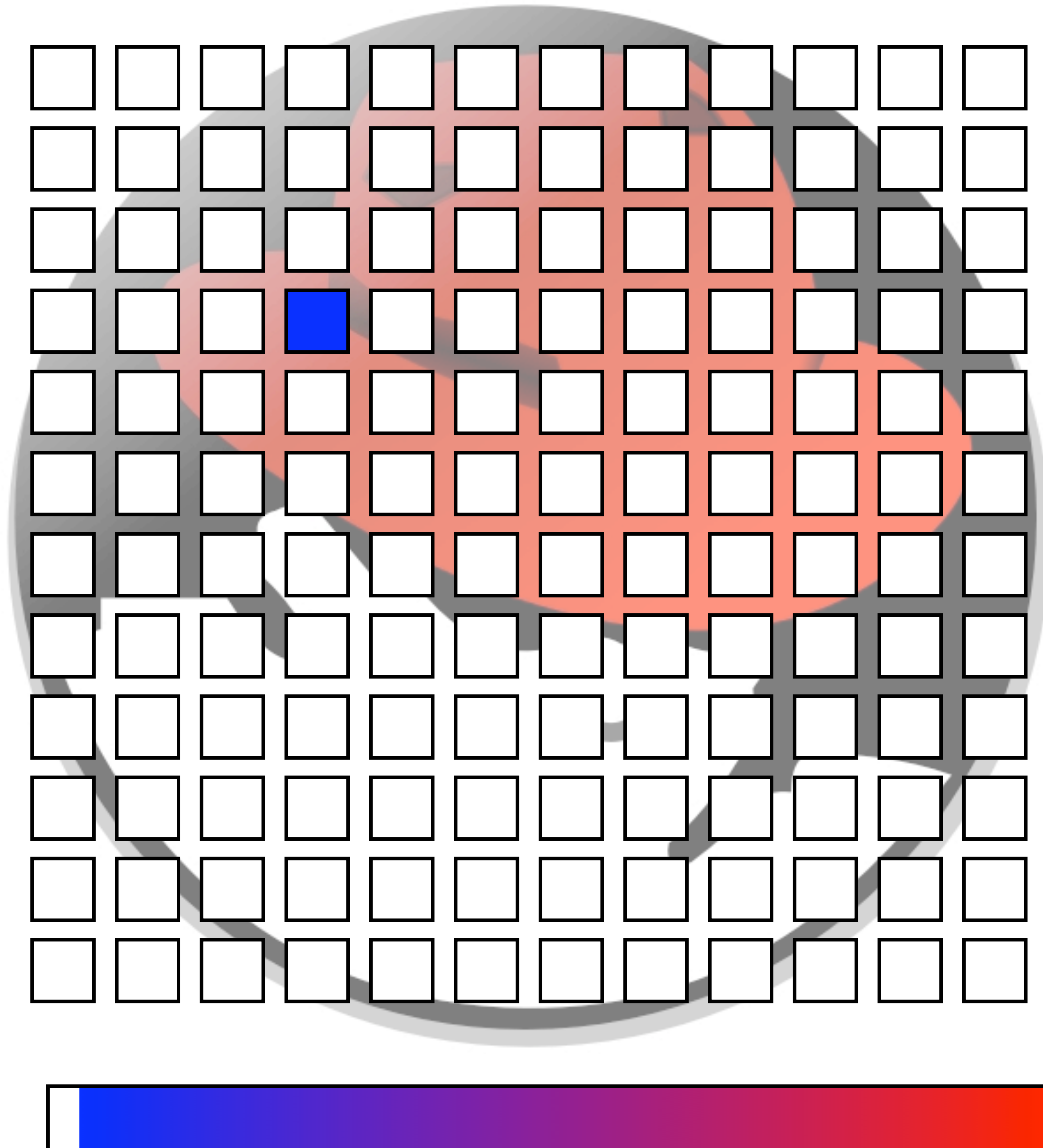
Explain colours: white = no vulnerabilities, blue -> red: progressively more

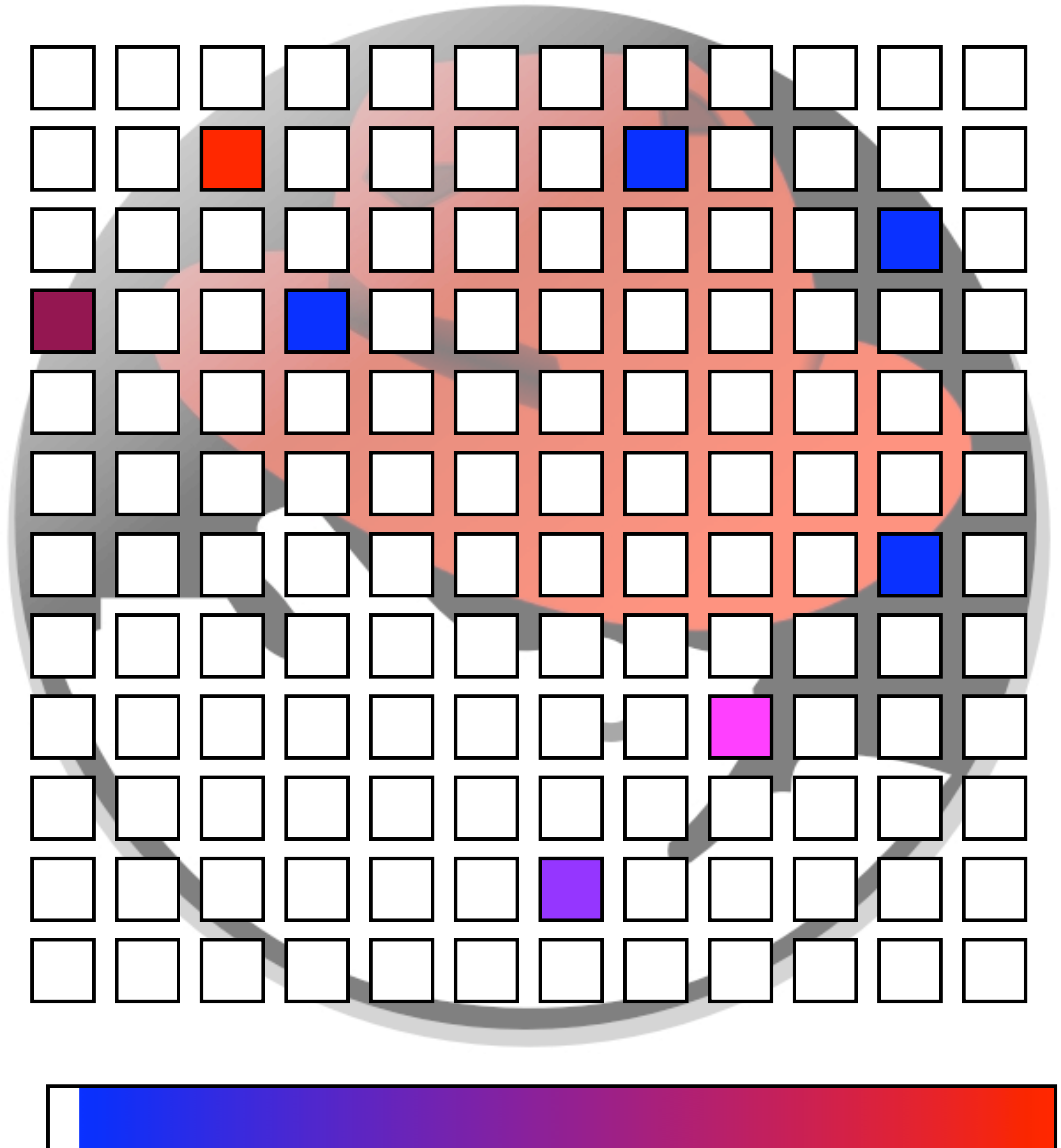


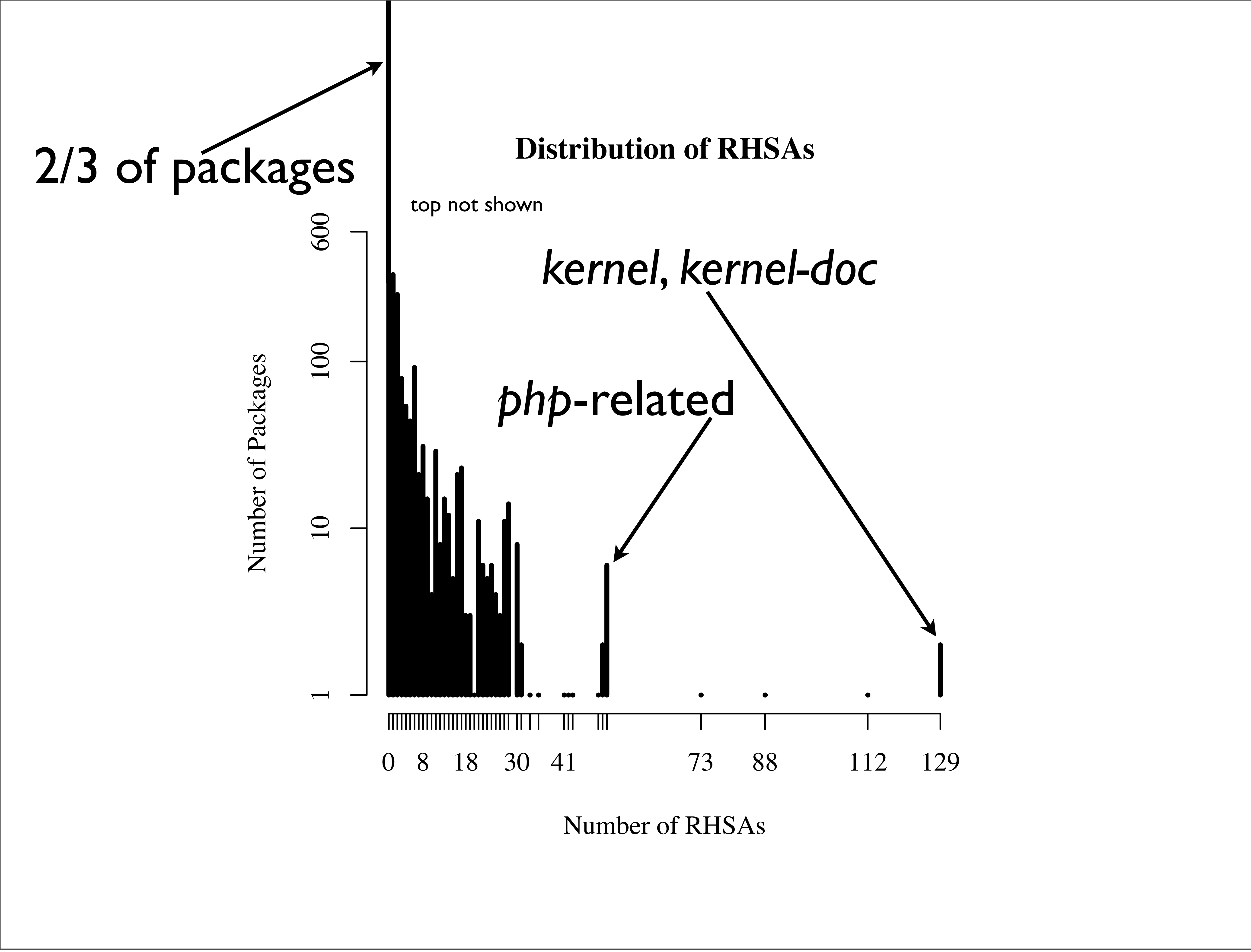
Explain colours: white = no vulnerabilities, blue -> red: progressively more



Explain colours: white = no vulnerabilities, blue -> red: progressively more







Note logarithmic y-axis. 3241 packages in total, about 2/3 with no known vulnerabilities.

Properties of packages, not properties of the software in the package

Are there properties that correlate with vulnerabilities?

Properties of packages, not properties of the software in the package

Are there properties that correlate with vulnerabilities?

Are there properties that increase or decrease the risk?

Properties of packages, not properties of the software in the package

Are there properties that correlate with vulnerabilities?

Are there properties that increase or decrease the risk?

Can we predict whether a package contains unknown vulnerabilities?

Properties of packages, not properties of the software in the package

Are there properties that correlate with vulnerabilities?

✔ Dependencies

Are there properties that increase or decrease the risk?

Can we predict whether a package contains unknown vulnerabilities?

Properties of packages, not properties of the software in the package

Are there properties that correlate with vulnerabilities?

✔ Dependencies

Are there properties that increase or decrease the risk?

✔ Beauties and Beasts

Can we predict whether a package contains unknown vulnerabilities?

Properties of packages, not properties of the software in the package

Are there properties that correlate with vulnerabilities?	✔ Dependencies
Are there properties that increase or decrease the risk?	✔ Beauties and Beasts
Can we predict whether a package contains unknown vulnerabilities?	✔ Machine Learning

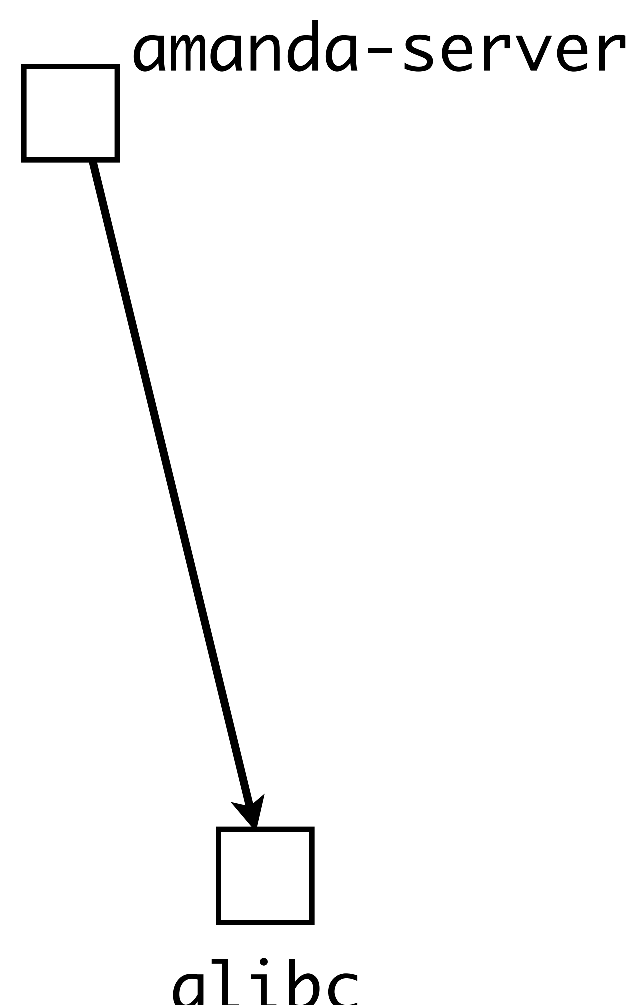
Properties of packages, not properties of the software in the package

Dependencies

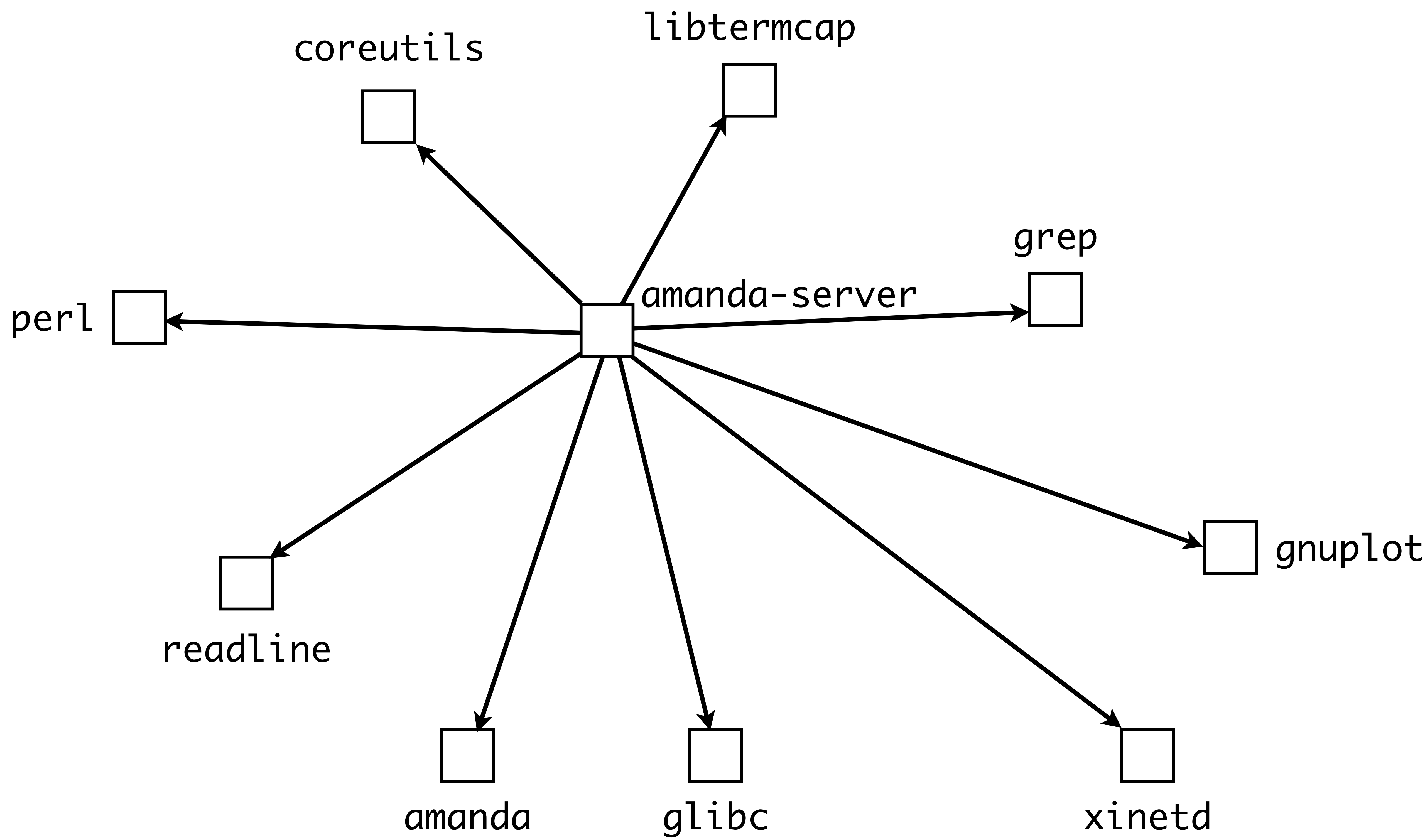
Dependencies

amanda-server

Dependencies



Dependencies

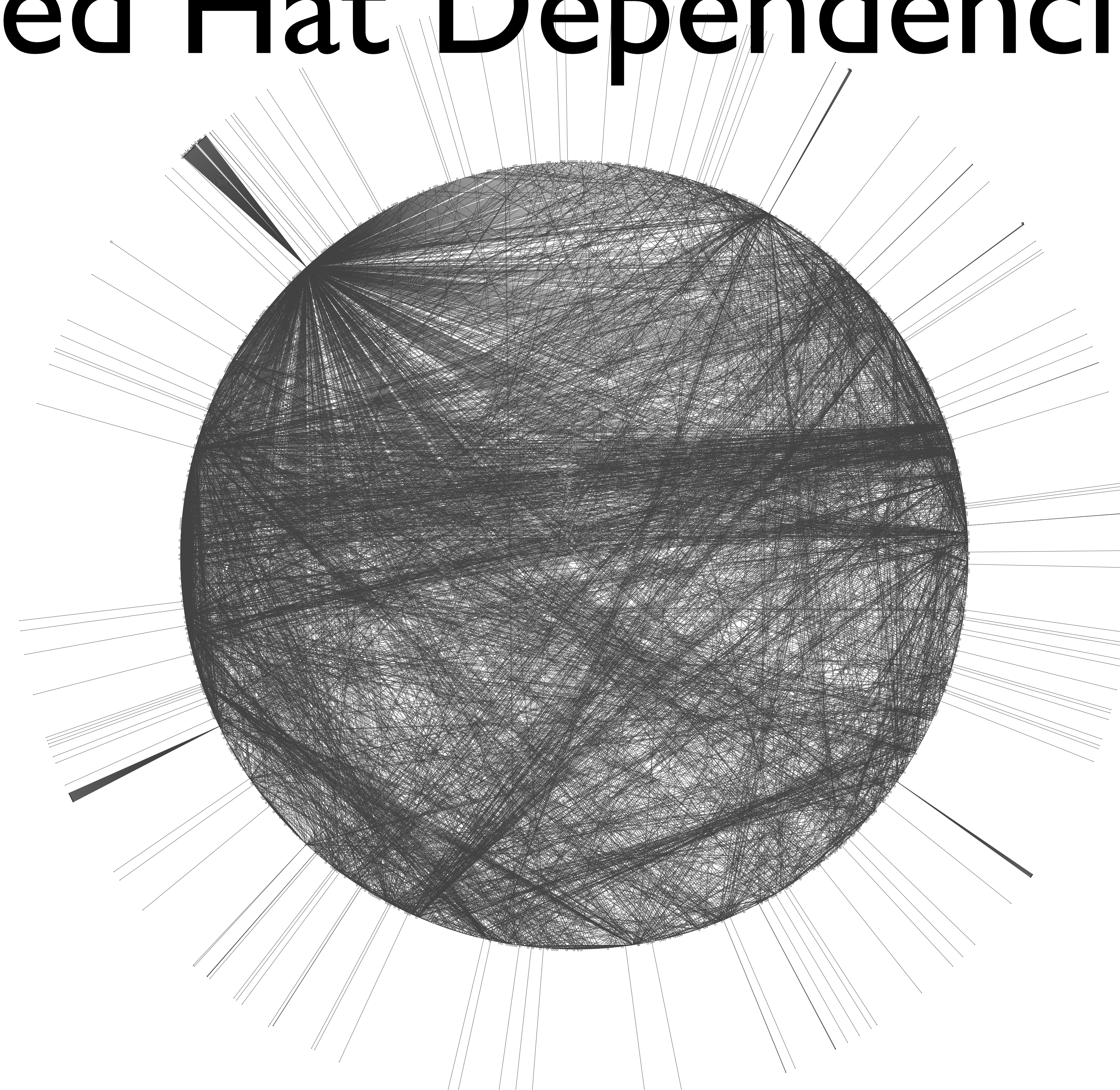


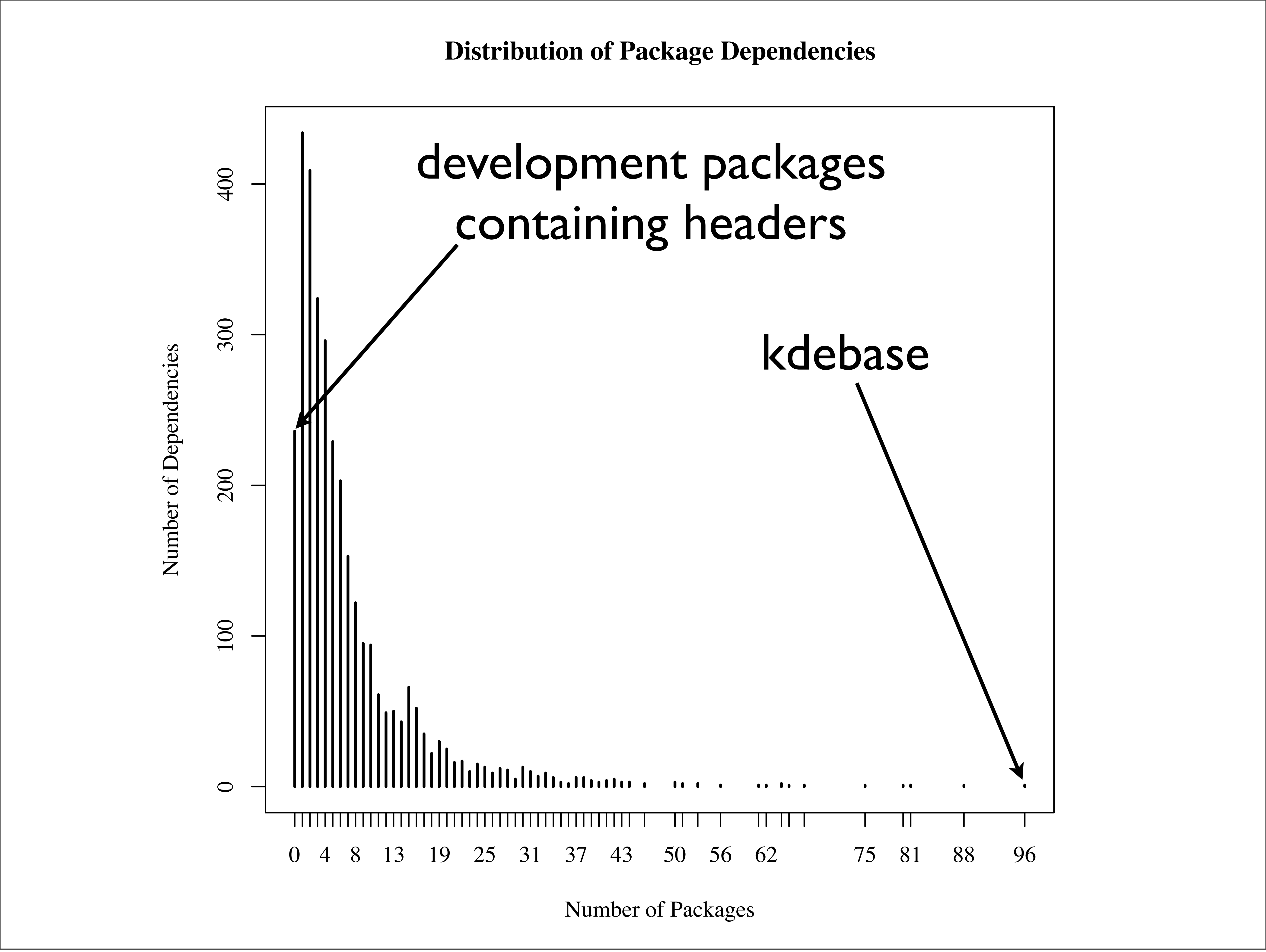
Dependencies and Vulnerabilities

- Dependency A → B exists because A wants to use the services offered by B
- Vulnerability exists in A if
  - A is in an *insecure domain* (domains are characterised by dependencies)
  - B is insecure and *fix* in B *spills over* to A; or
  - B is *difficult to use securely*

Packages in same domain will tend to have same dependencies.
Domain examples are: compilers, games, office applications,

Red Hat Dependencies





Distribution is apparently logarithmic with a long tail. This is not transitive closure. kdebase has 14 RHSAs (but 96 dependencies), kernel has 129 (but 0 dependencies), so number of dependencies is not a good predictor of number of RHSAs

Are there properties that correlate with vulnerabilities?	✔ Dependencies
Are there properties that increase or decrease the risk?	✔ Beauties and Beasts
Can we predict whether a package contains unknown vulnerabilities?	✔ Machine Learning

Where does the addition of dependencies significantly increase/ decrease the risk?

Where does the addition of dependencies significantly increase/ decrease the risk?

1. Data structure: concept lattice

Where does the addition of dependencies significantly increase/ decrease the risk?

1. Data structure: concept lattice
2. Compute change in risk

Where does the addition of dependencies significantly increase/decrease the risk?

1. Data structure: concept lattice
2. Compute change in risk
3. Include only statistically significant changes

Step 1: Data Structure

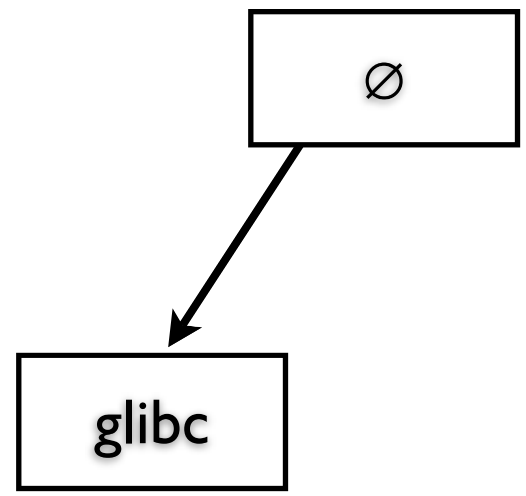
Start with no knowledge about dependencies (top node contains all packages). Add knowledge of glibc (node contains all packages depending on glibc), then qt (node contains all packages depending on qt and glibc), then xorg-x11-libs (node contains all packages

Step 1: Data Structure

$\boxed{\varnothing}$

Start with no knowledge about dependencies (top node contains all packages). Add knowledge of glibc (node contains all packages depending on glibc), then qt (node contains all packages depending on qt and glibc), then xorg-x11-libs (node contains all packages

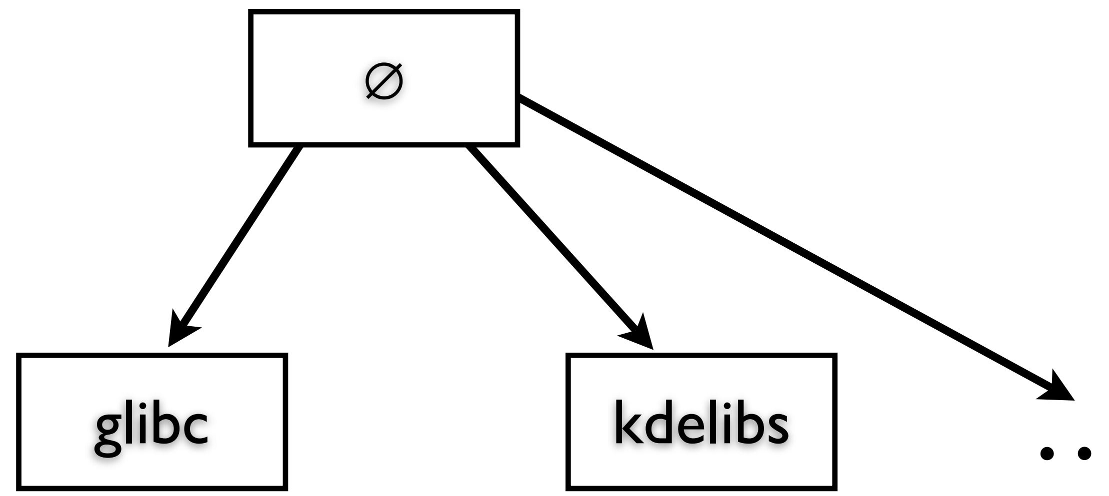
Step 1: Data Structure



Block 1: All packages depending on *glibc*

Start with no knowledge about dependencies (top node contains all packages). Add knowledge of glibc (node contains all packages depending on glibc), then qt (node contains all packages depending on qt and glibc), then xorg-x11-libs (node contains all packages

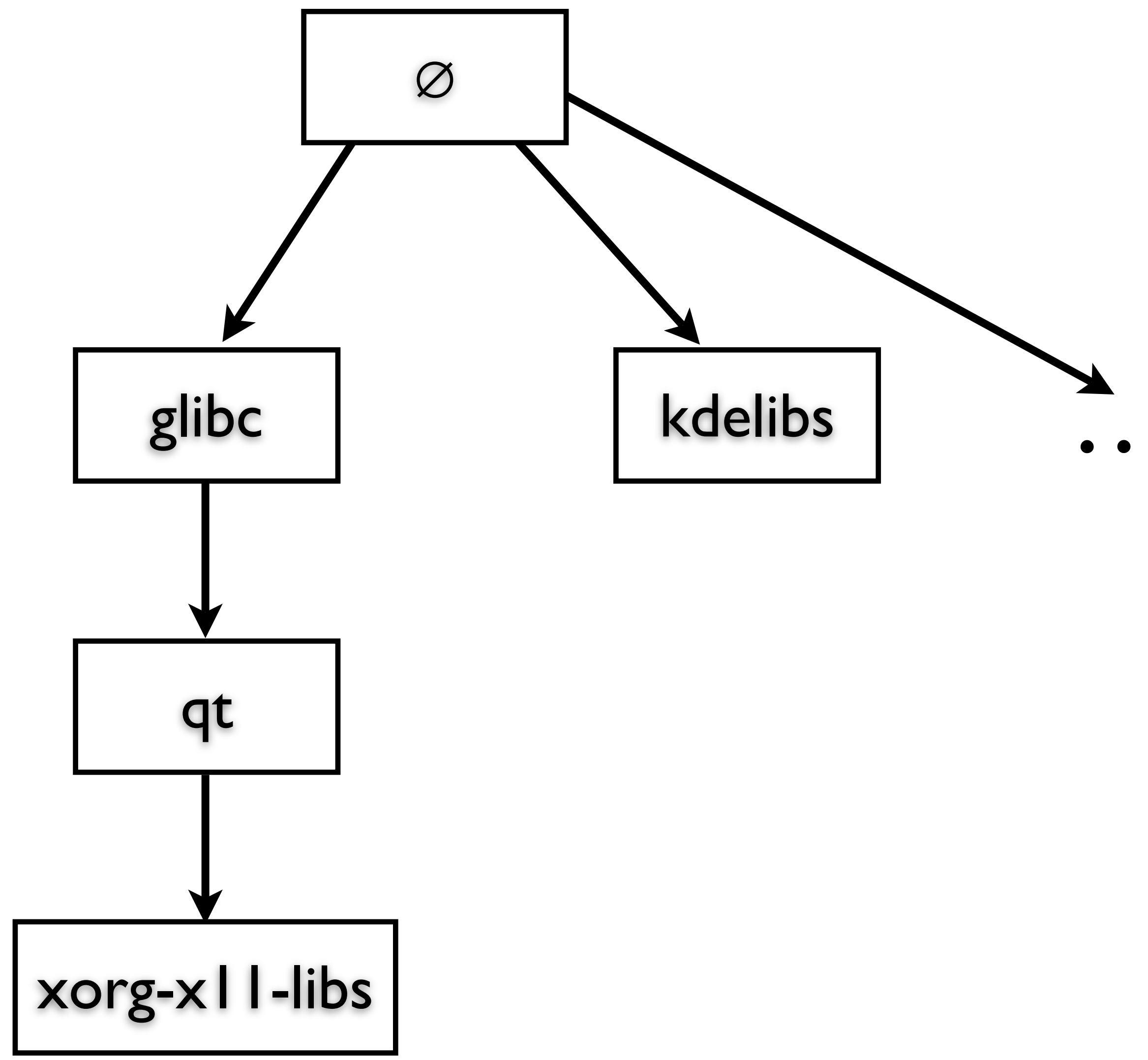
Step 1: Data Structure



Block 1: All packages depending on *glibc*

Start with no knowledge about dependencies (top node contains all packages). Add knowledge of glibc (node contains all packages depending on glibc), then qt (node contains all packages depending on qt and glibc), then xorg-x11-libs (node contains all packages

Step 1: Data Structure



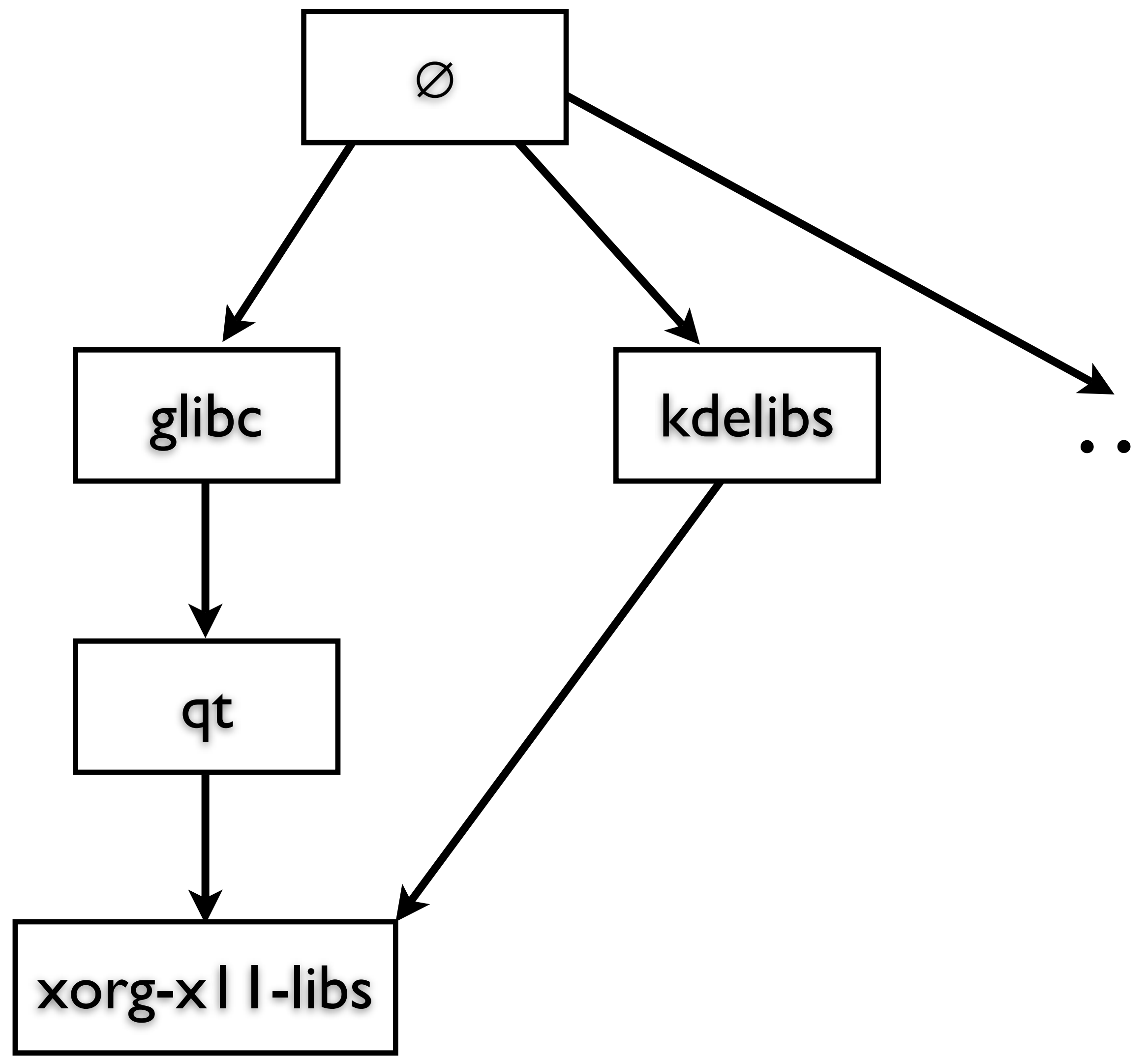
Block 1: All packages depending on *glibc*

Block 2: All packages depending on *glibc, qt*

Block 3: All packages depending on *glibc, qt, xorg-x11-libs*

Start with no knowledge about dependencies (top node contains all packages). Add knowledge of glibc (node contains all packages depending on glibc), then qt (node contains all packages depending on qt and glibc), then xorg-x11-libs (node contains all packages

Step 1: Data Structure



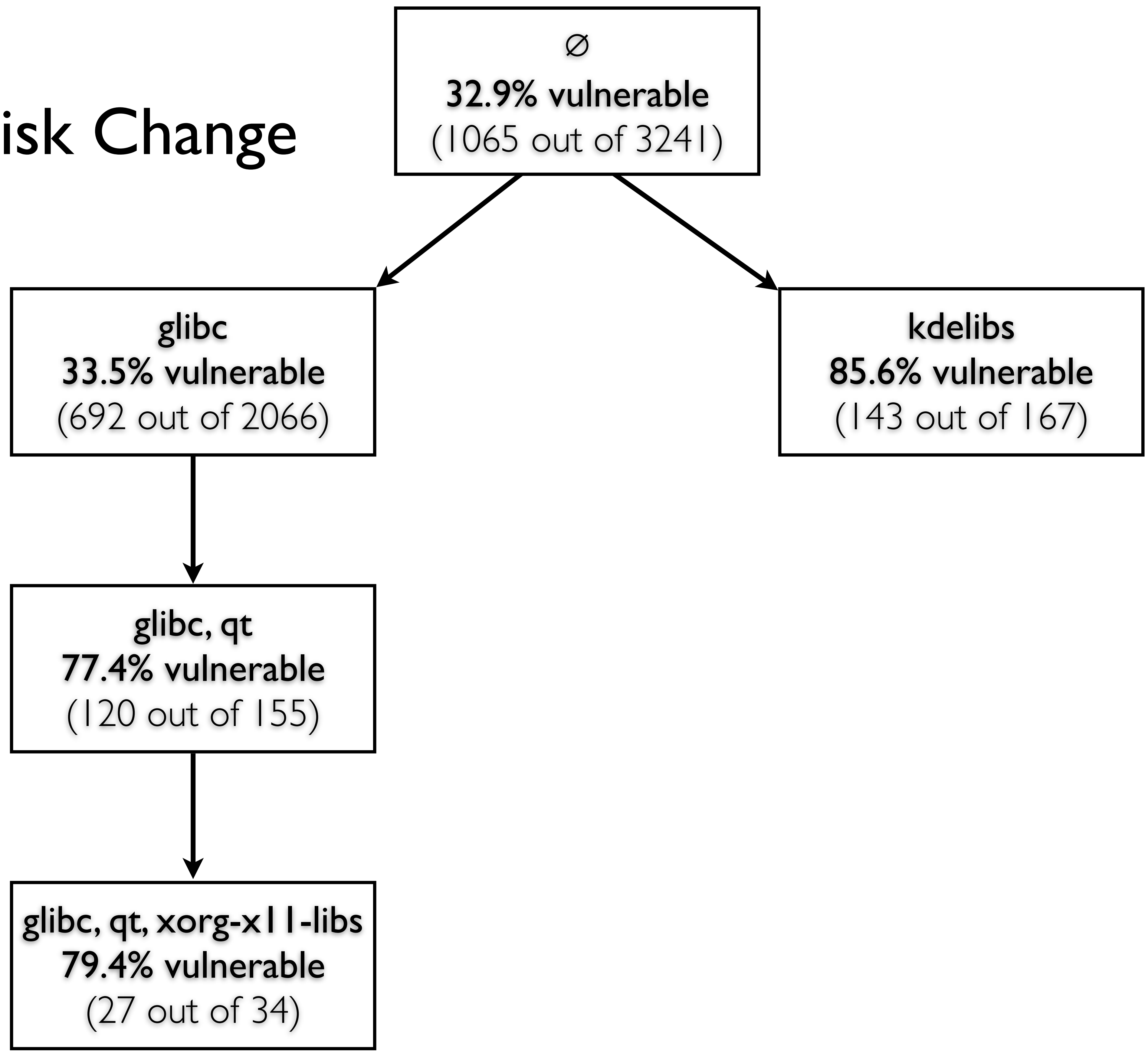
Block 1: All packages depending on *glibc*

Block 2: All packages depending on *glibc, qt*

Block 3: All packages depending on *glibc, qt, xorg-x11-libs*

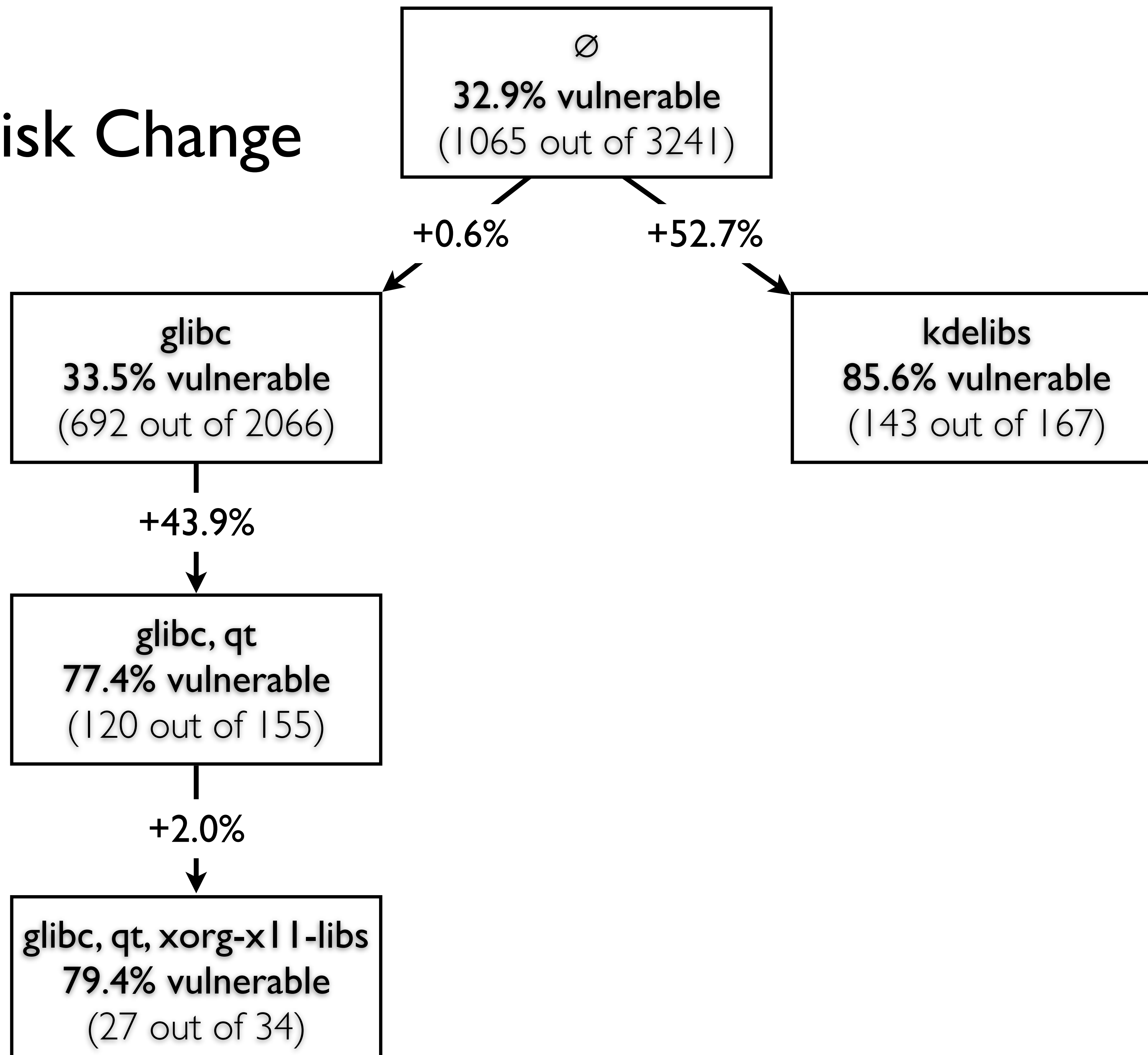
Start with no knowledge about dependencies (top node contains all packages). Add knowledge of glibc (node contains all packages depending on glibc), then qt (node contains all packages depending on qt and glibc), then xorg-x11-libs (node contains all packages

Step 2:
Compute Risk Change



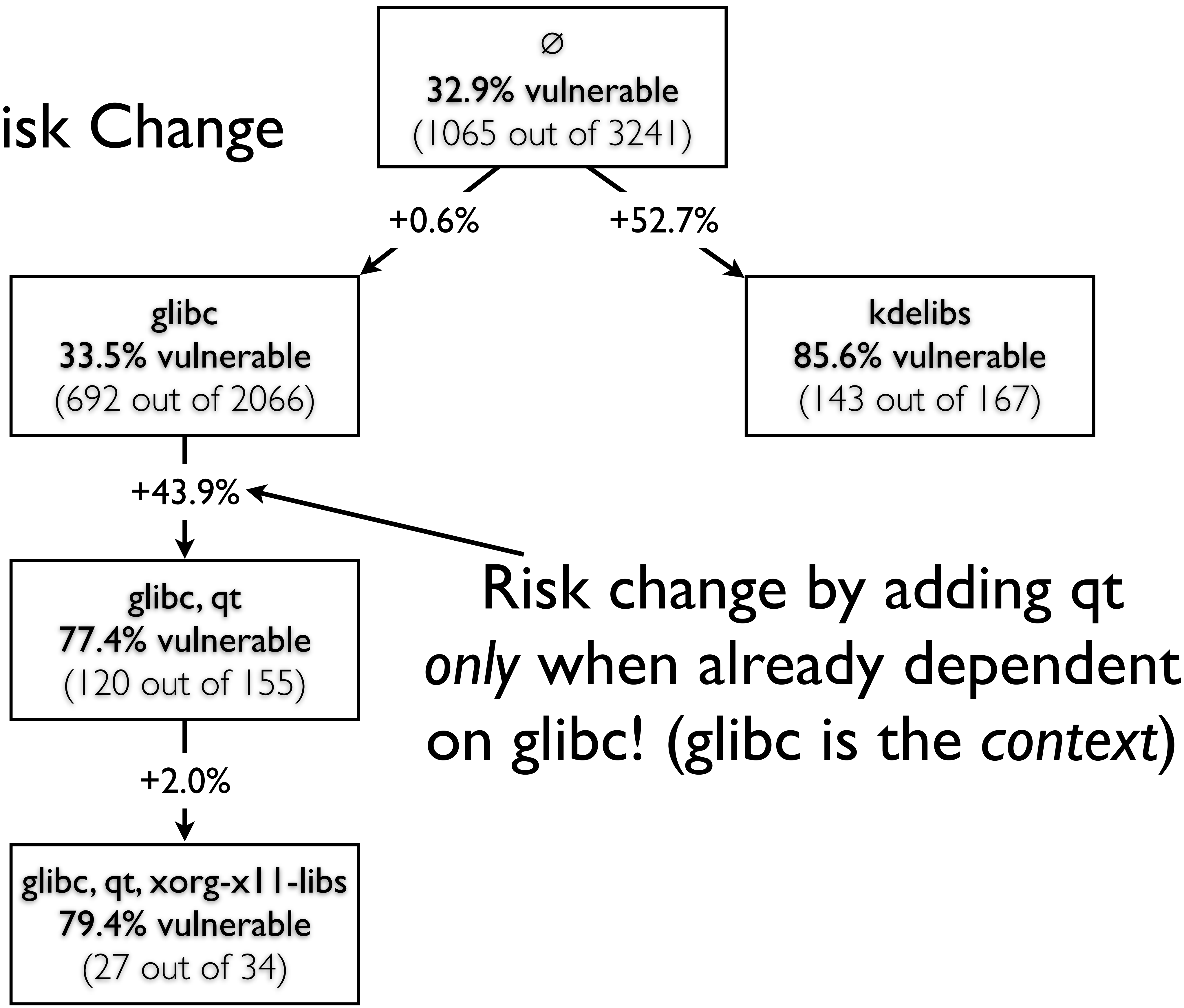
Question: Is the rise of 43.9% when going from {glibc} to {glibc, qt} just some random fluctuation? We test this using statistical tests (Chi^2 or Fischer exact) and discard the “random fluctuation” hypothesis when the probability of such a increase happening

Step 2:
Compute Risk Change



Question: Is the rise of 43.9% when going from {glibc} to {glibc, qt} just some random fluctuation? We test this using statistical tests (Chi^2 or Fischer exact) and discard the “random fluctuation” hypothesis when the probability of such a increase happening

Step 2:
Compute Risk Change



Question: Is the rise of 43.9% when going from {glibc} to {glibc, qt} just some random fluctuation? We test this using statistical tests (Chi^2 or Fischer exact) and discard the “random fluctuation” hypothesis when the probability of such a increase happening

Step 3: Include Only Significant Changes

- Risk changes with significance $p < 0.01$
- No significant and more general context exists for this dependency
- Risk goes up: “beast”
- Risk goes down: “beauty”

Selected Beasts

The complete list can be found in the paper

Context	Dependency	Risk before	Risk after	Change
∅	openoffice.org-core	0.329	1.000	0.671
∅	kdelibs	0.329	0.856	0.527
∅	cups-libs	0.329	0.774	0.445
∅	libmng	0.329	0.769	0.440
glibc	qt	0.335	0.774	0.439
glibc	krb5-libs	0.335	0.769	0.434

Explain packages, don’t just list names

Selected Beauties

The complete list can be found in the paper

Context	Dependency	Risk before	Risk after	Change
glibc	xorg-x11-server-Xorg	0.335	0.015	-0.320
compat-glibc, glibc, zlib	audiofile	0.613	0.359	-0.254
glibc, glibc-debug, zlib	audiofile	0.590	0.351	-0.239
∅	gnome-keyring	0.329	0.101	-0.228
glibc, zlib	gnome-libs	0.456	0.281	-0.175
∅	python	0.329	0.132	-0.197

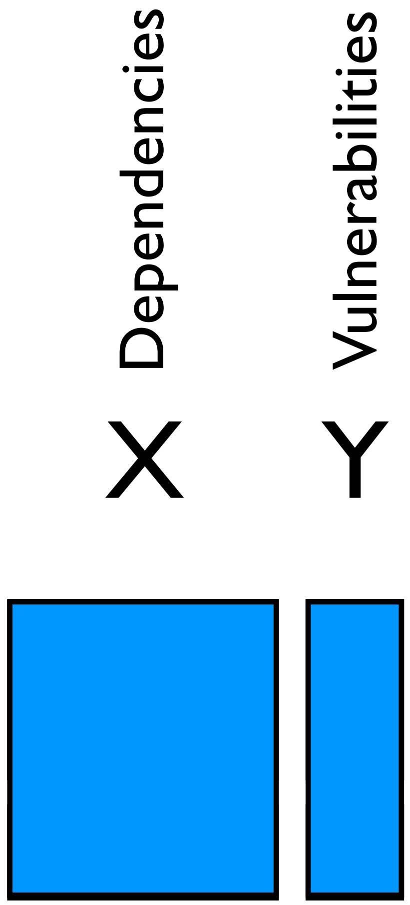
Explain possible consequences: new applications: choose less risky dependencies

Are there properties that correlate with vulnerabilities?	✔ Dependencies
Are there properties that increase or decrease the risk?	✔ Beauties and Beasts
Can we predict whether a package contains unknown vulnerabilities?	✔ Machine Learning

Is it possible to predict…

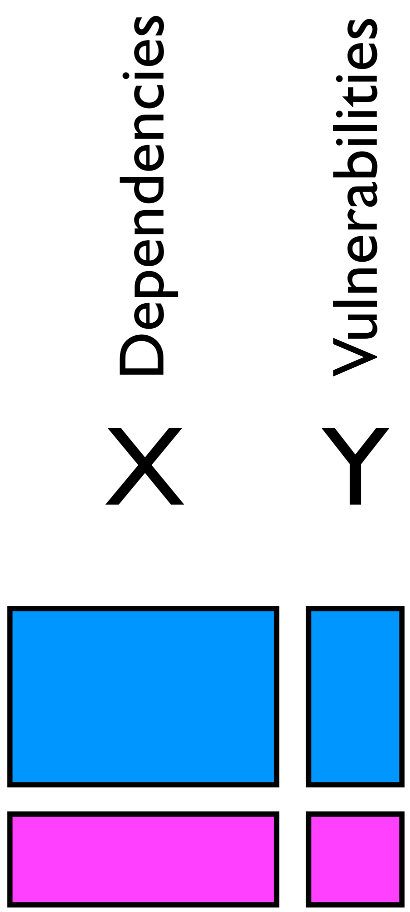
- from the dependencies *which packages are vulnerable* (classification)?
- *which packages* will have the *most vulnerabilities* (ranking)?

Experiment



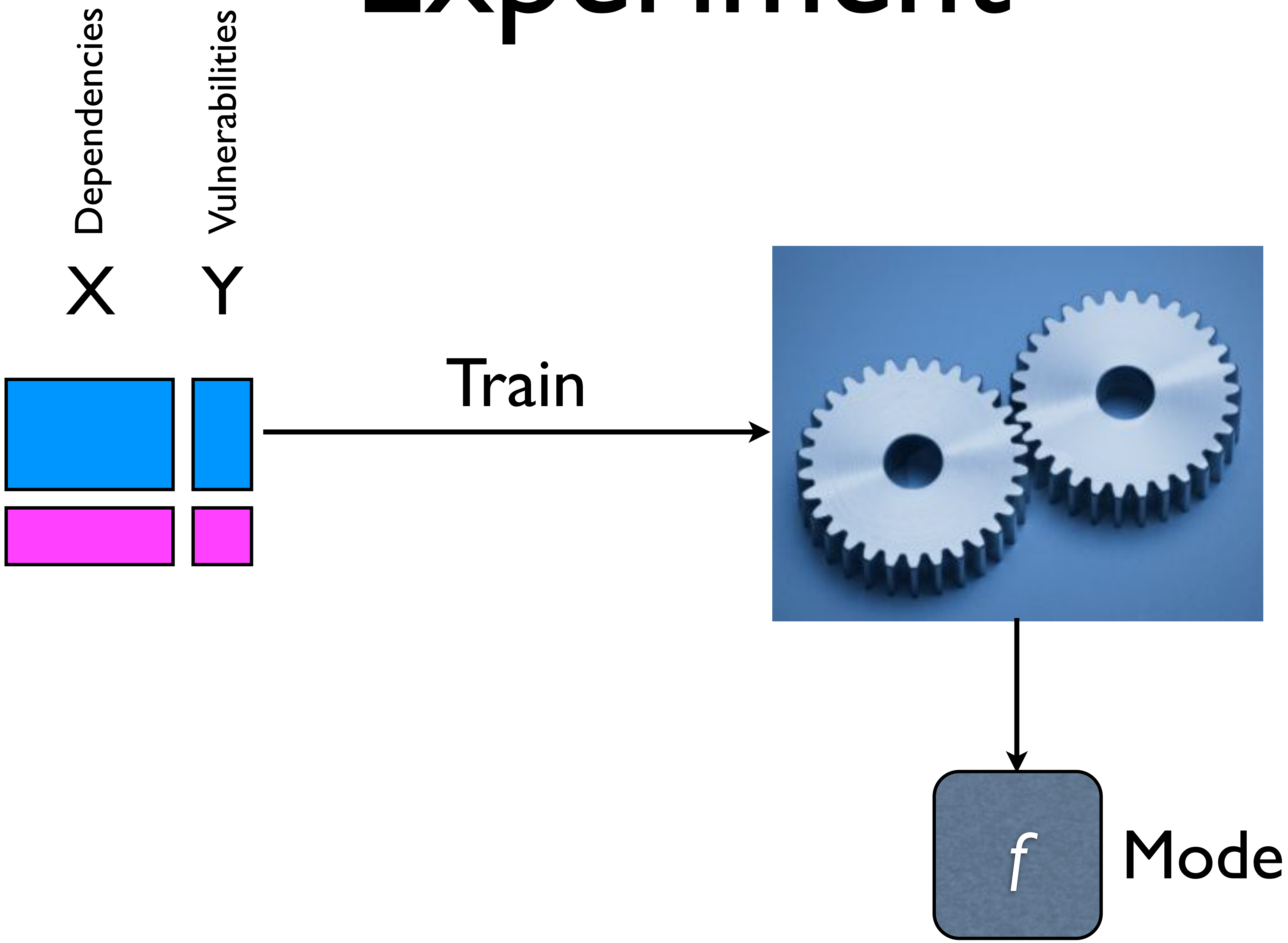
Repeat 50x
This “self-testing” is a standard evaluation technique for machine learning methods

Experiment



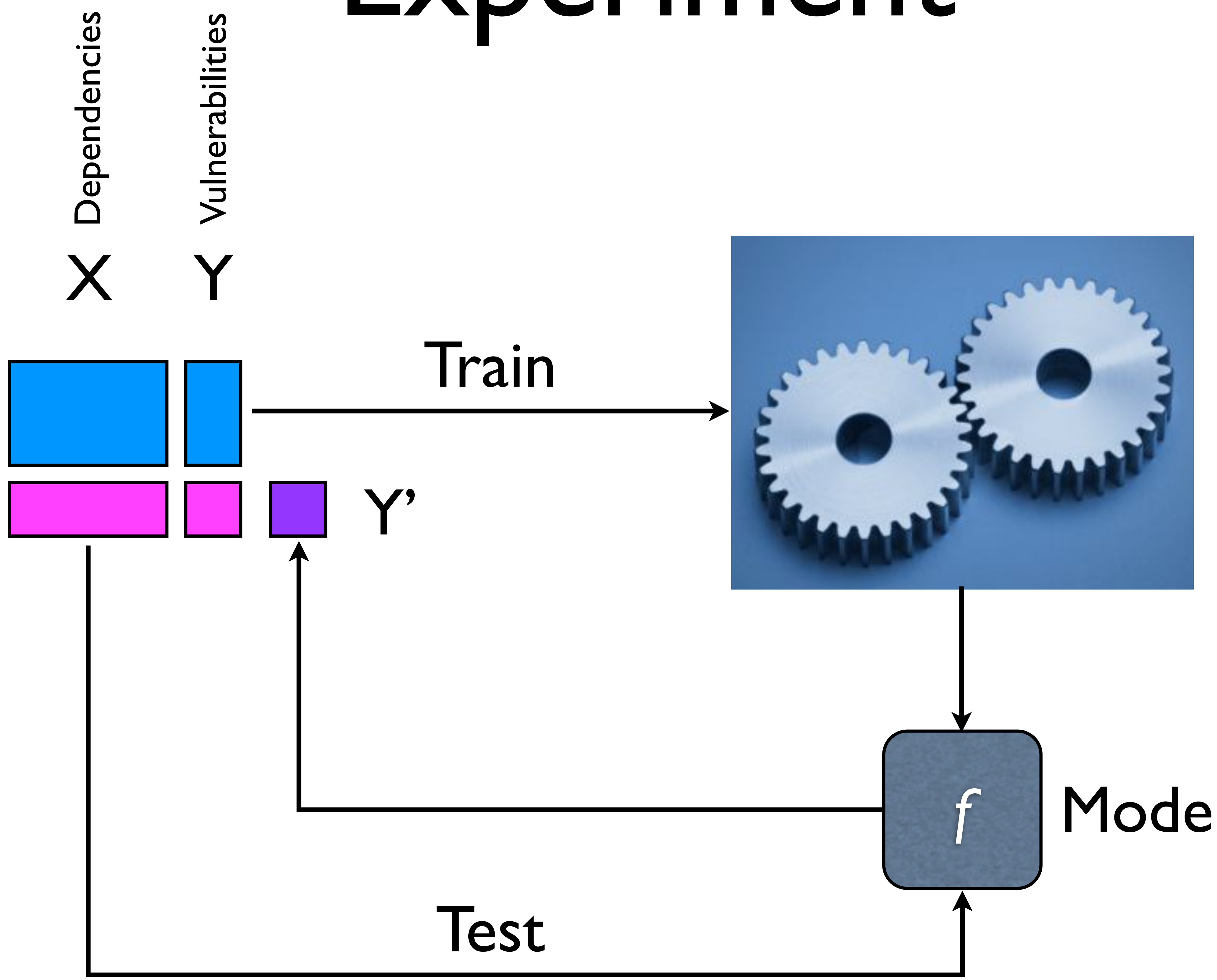
Repeat 50x
This “self-testing” is a standard evaluation technique for machine learning methods

Experiment



Repeat 50x
This “self-testing” is a standard evaluation technique for machine learning methods

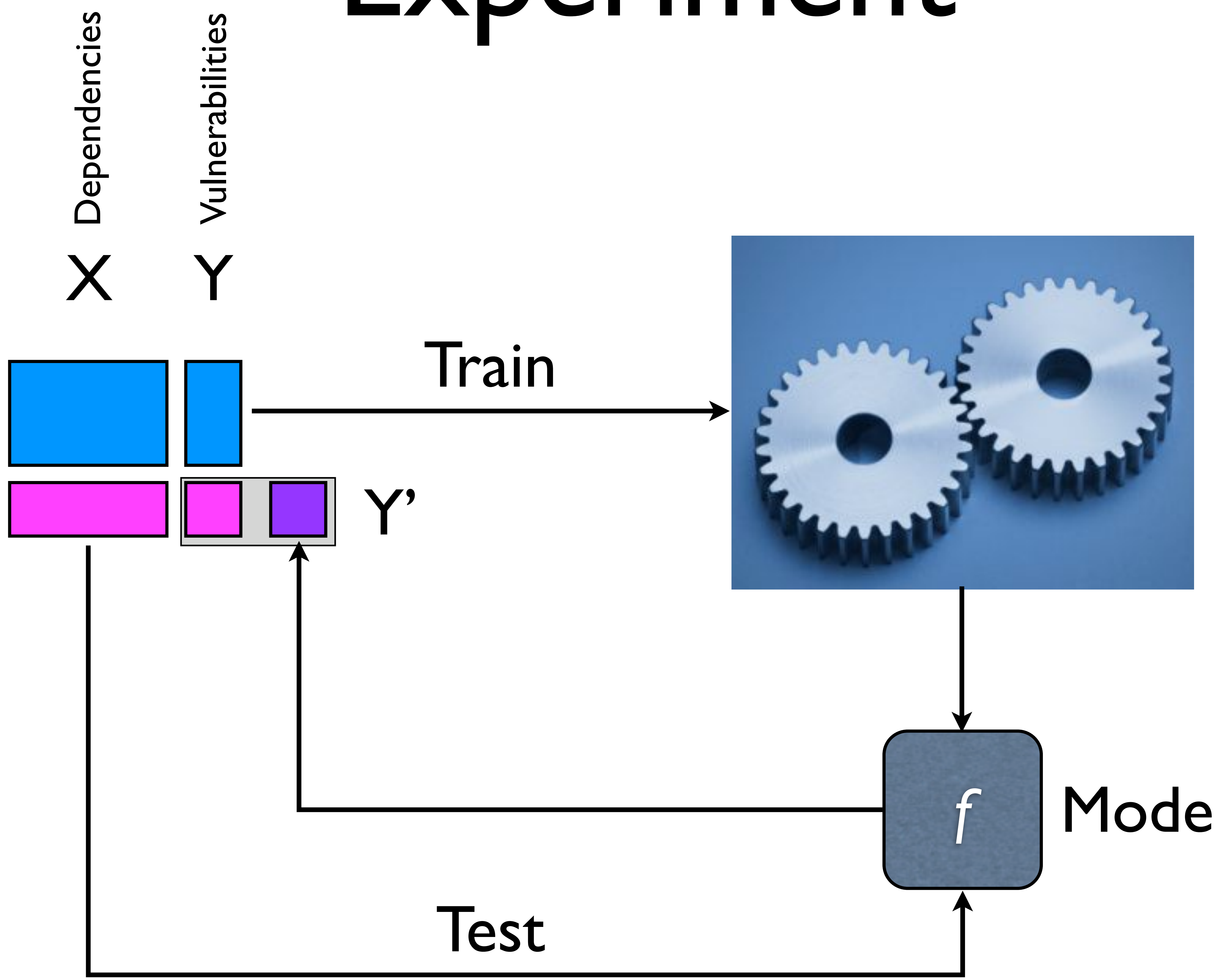
Experiment



Repeat 50x

This “self-testing” is a standard evaluation technique for machine learning methods

Experiment



Repeat 50x

This “self-testing” is a standard evaluation technique for machine learning methods

Indicators

Don't mention −1. We want values near 1.

Indicators

Classification

Don’t mention −1. We want values near 1.

Indicators

Classification

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Don’t mention −1. We want values near 1.

Indicators

Classification

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

1

0

Don't mention -1. We want values near 1.

Indicators

Classification

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

1

0

Ranking

Don’t mention −1. We want values near 1.

Indicators

Classification

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

1

↑

0

Ranking

$1 \longrightarrow 1$

$2 \longrightarrow 2$

$3 \longrightarrow 3$

$4 \longrightarrow 4$

1

Don’t mention −1. We want values near 1.

Indicators

Classification

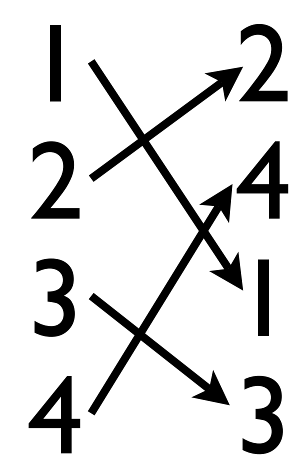
$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

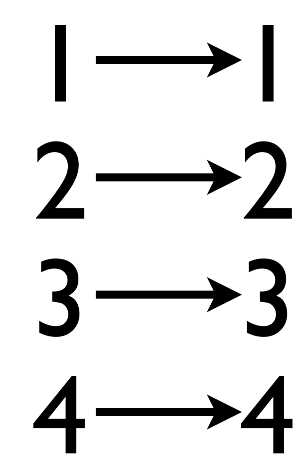
$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

1

0

Ranking





0

1

Don’t mention −1. We want values near 1.

Indicators

Classification

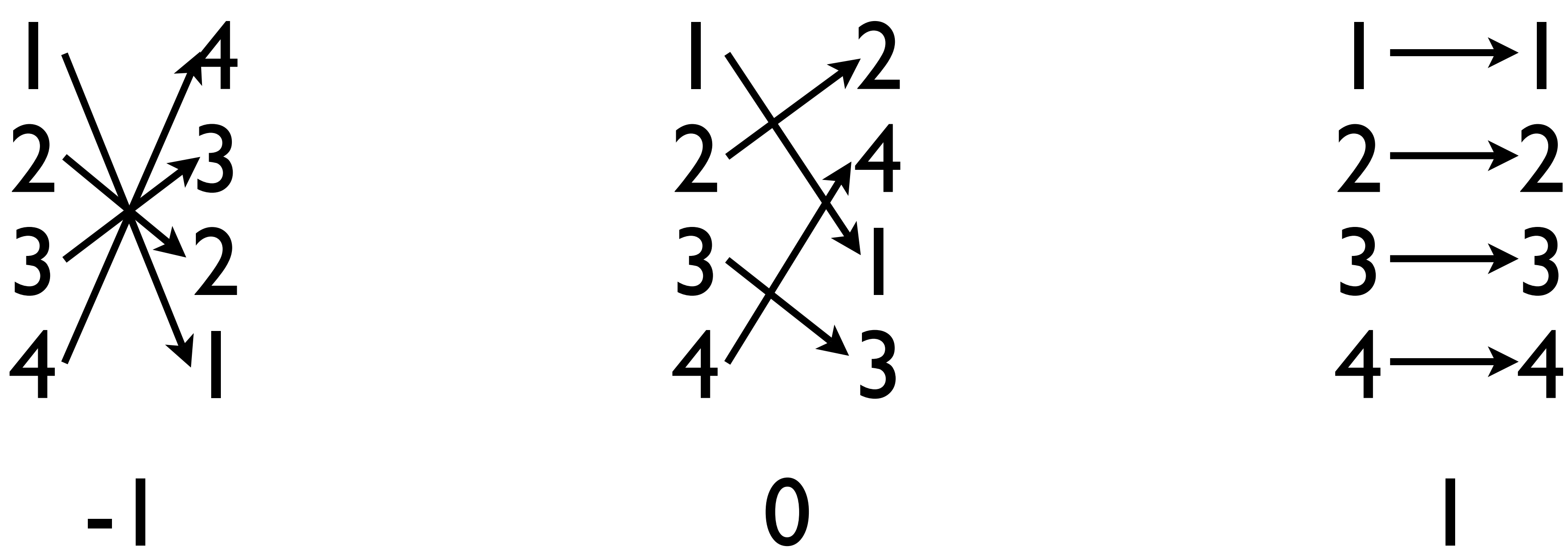
$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

1

0

Ranking



Don’t mention −1. We want values near 1.

Indicators

Classification

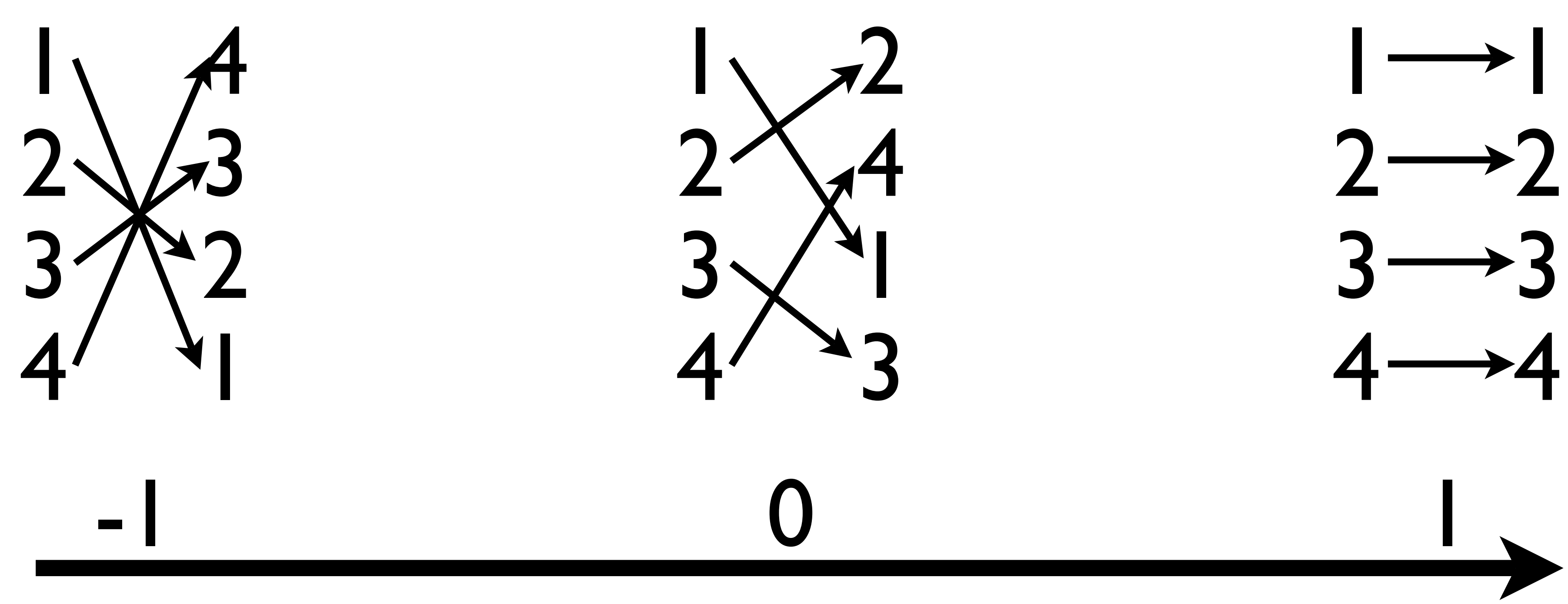
$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

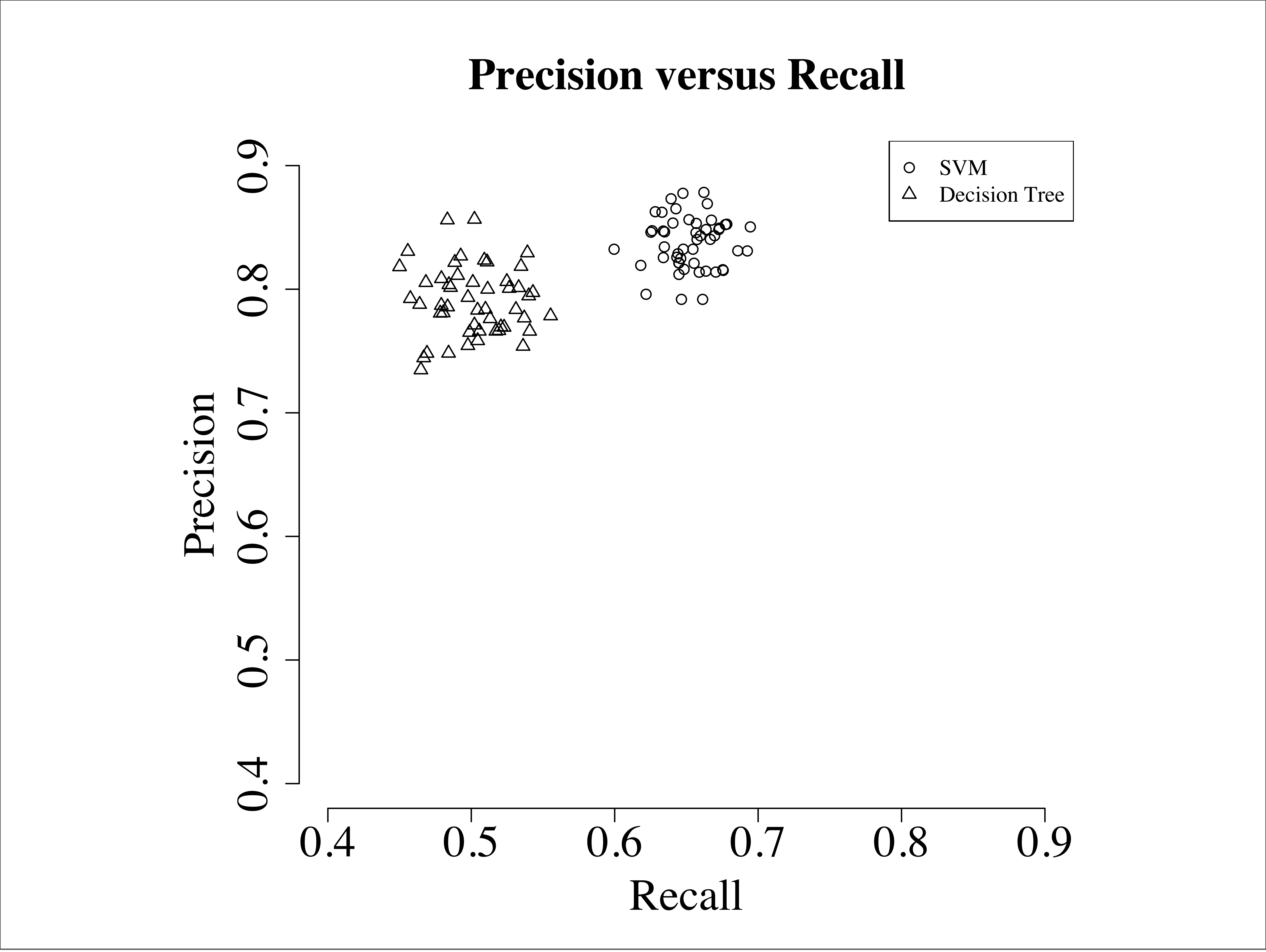
1

0

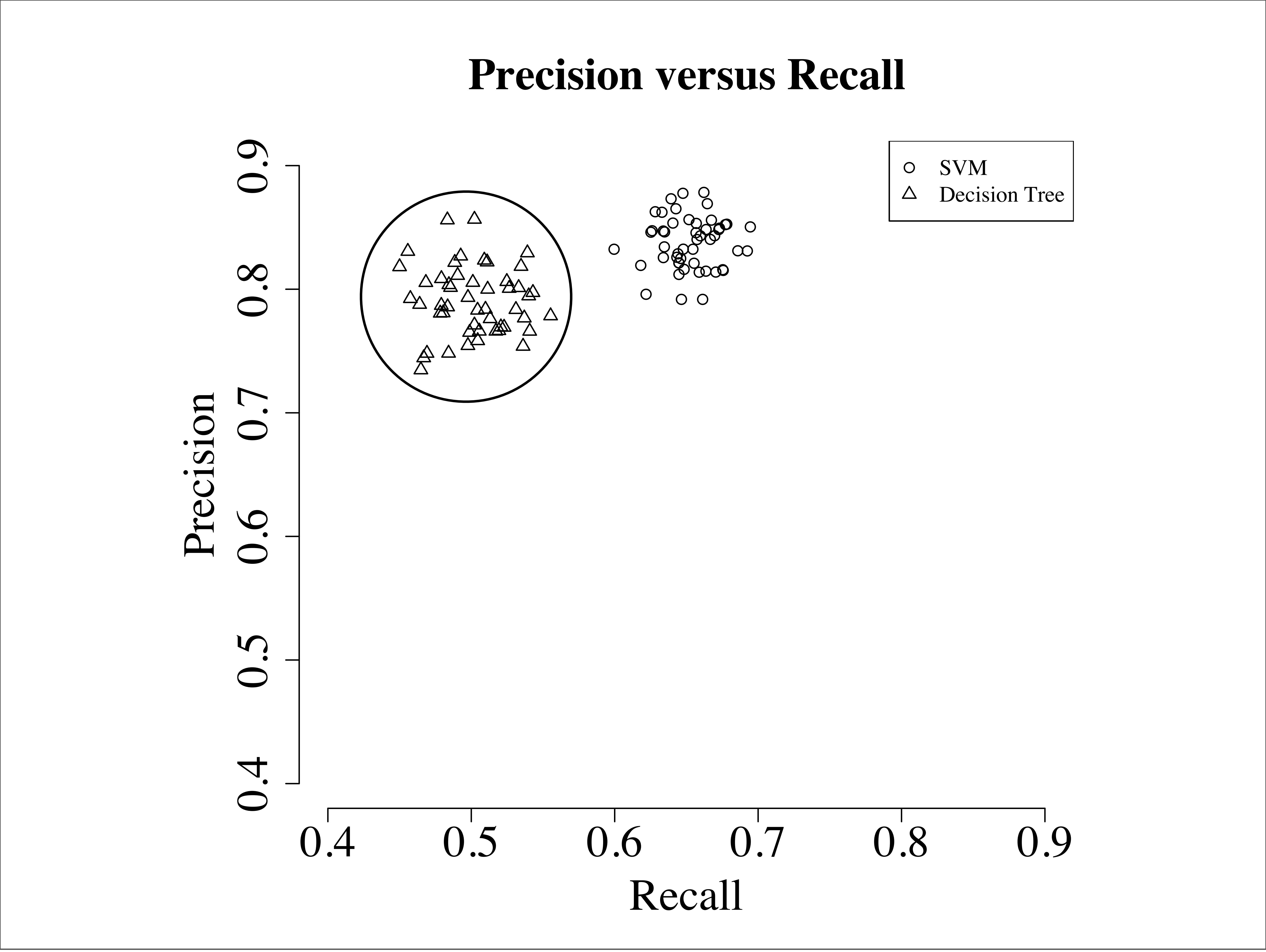
Ranking



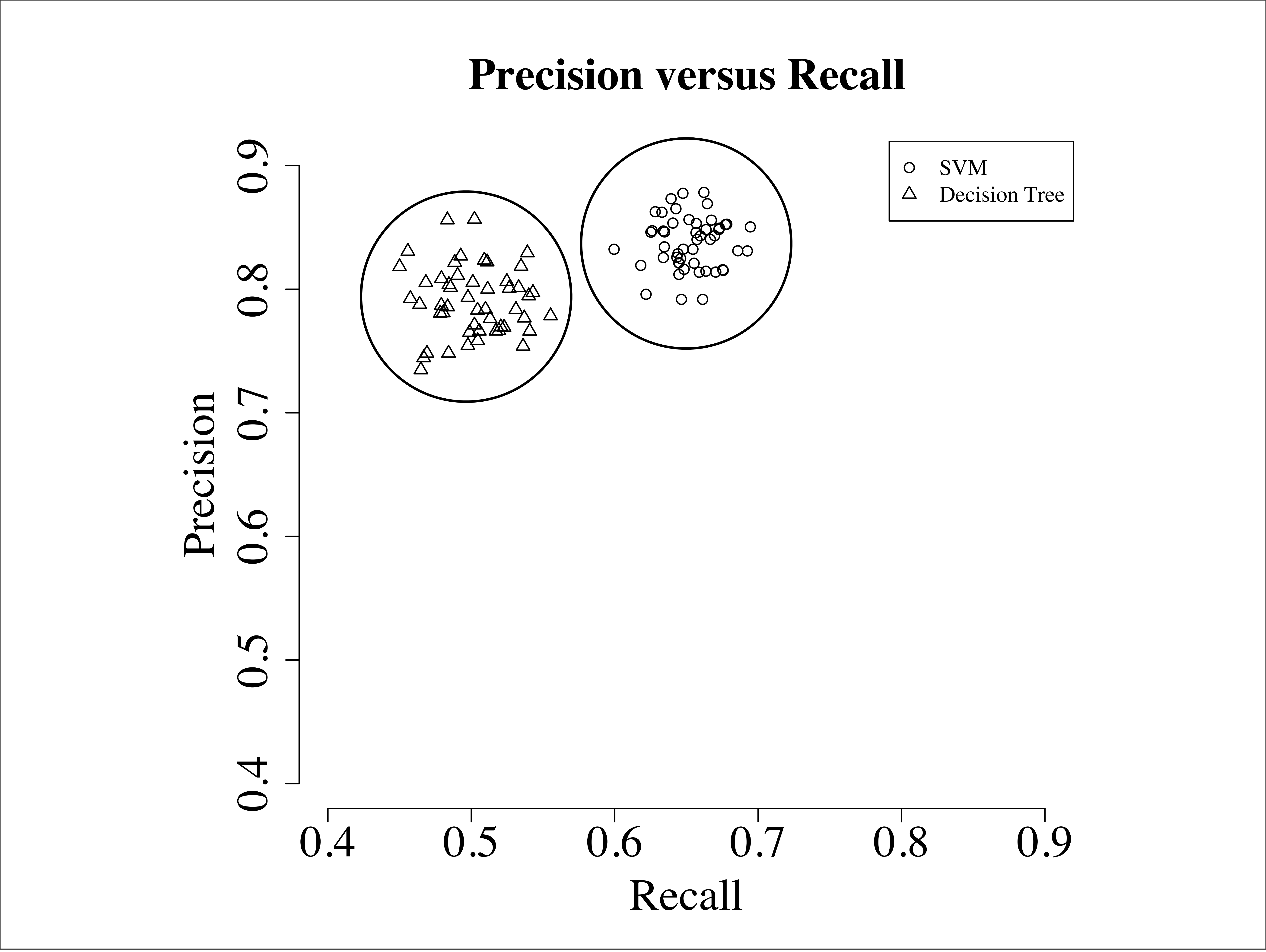
Don't mention −1. We want values near 1.



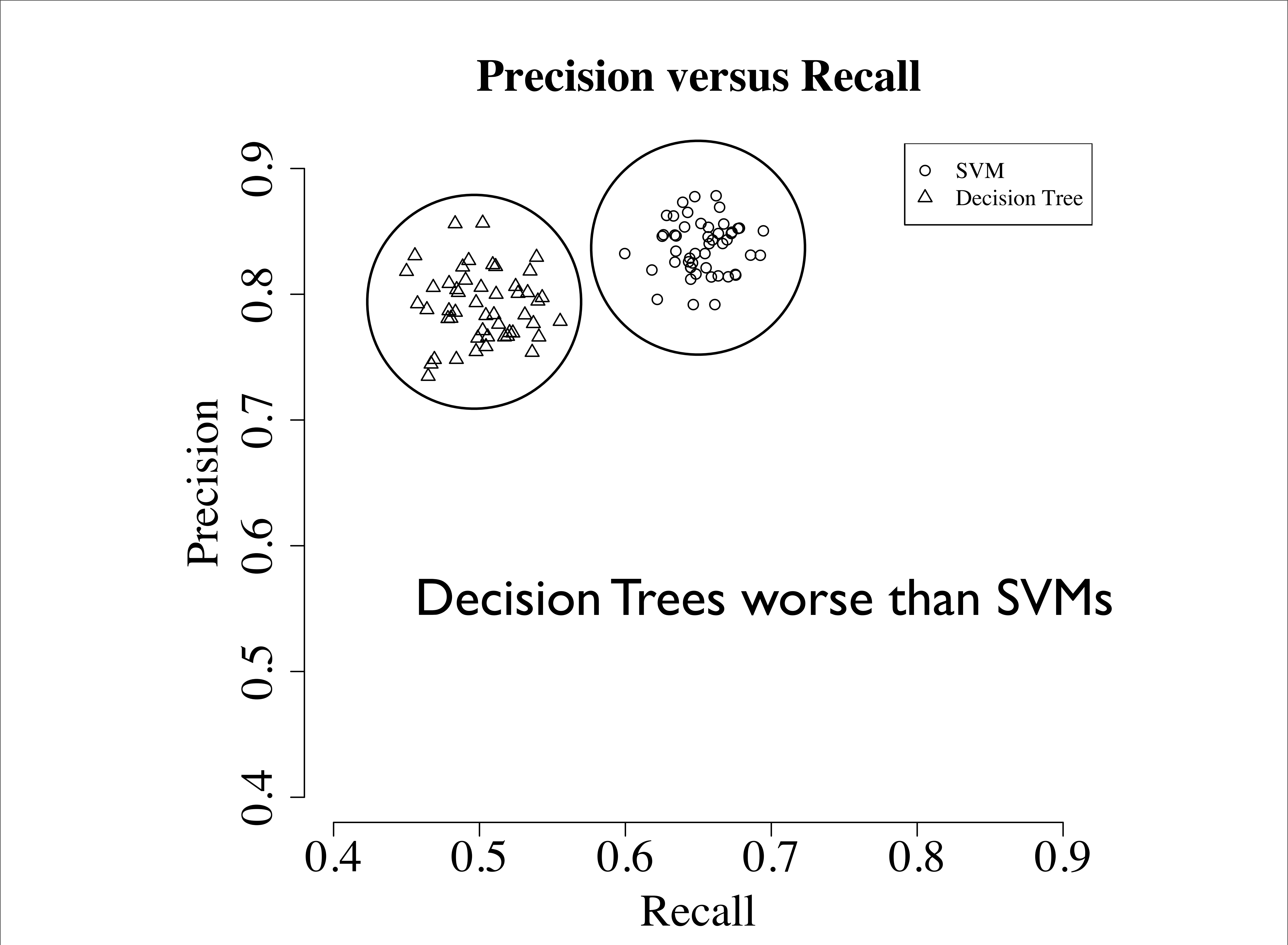
Results of 50 random splits: train with 2/3 of the packages, predict with the rest, record precision and recall.



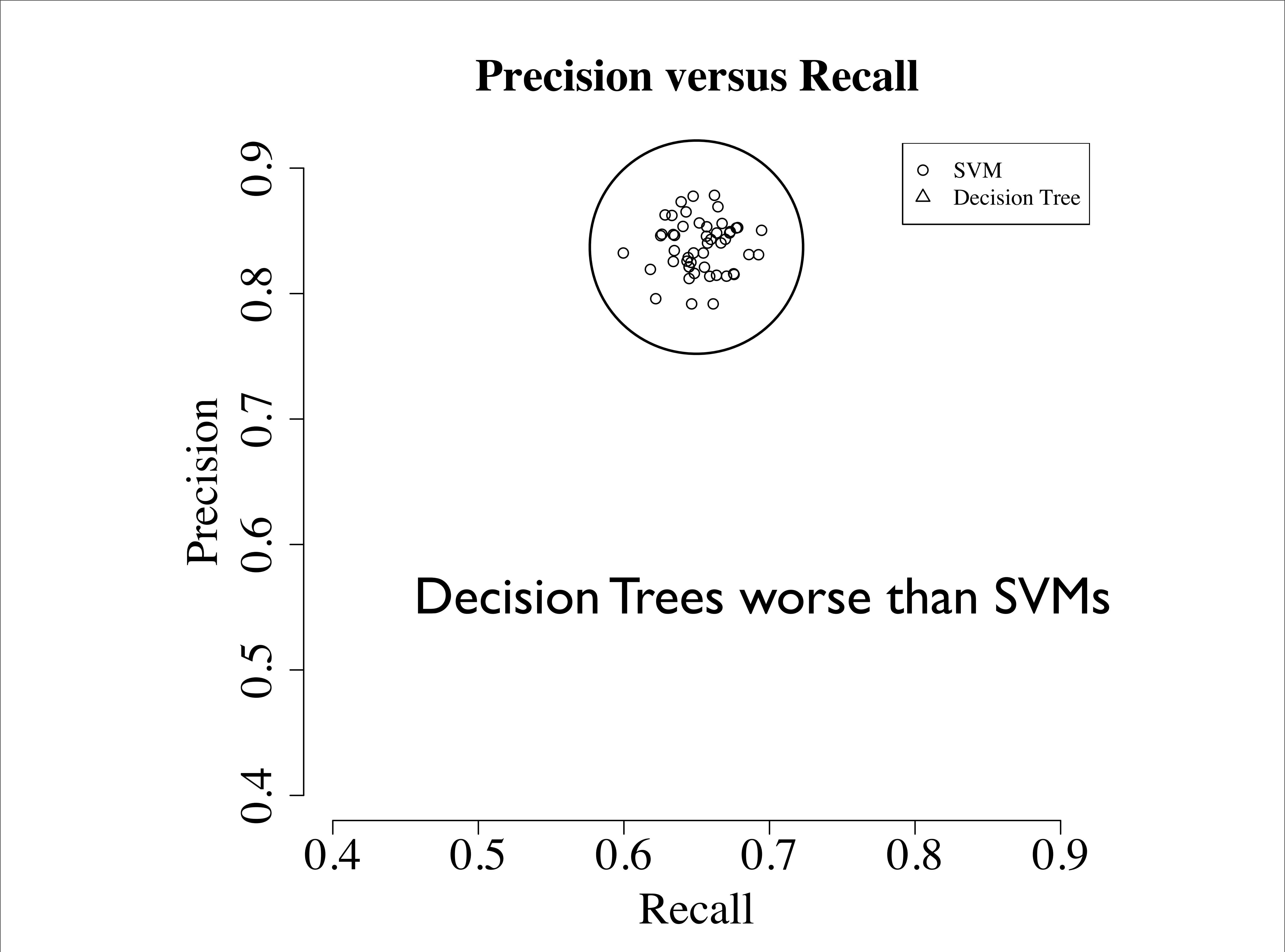
Results of 50 random splits: train with 2/3 of the packages, predict with the rest, record precision and recall.



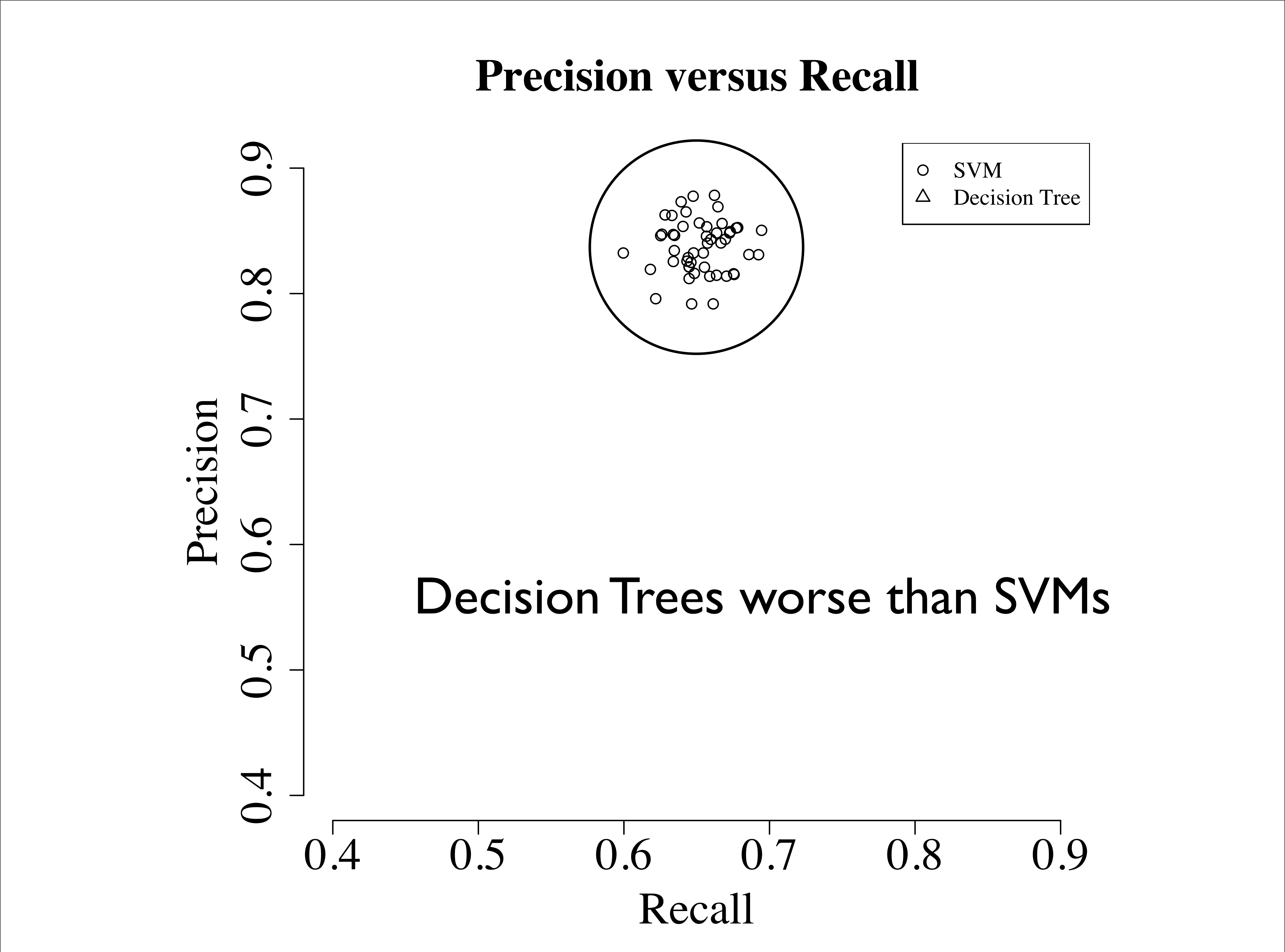
Results of 50 random splits: train with 2/3 of the packages, predict with the rest, record precision and recall.



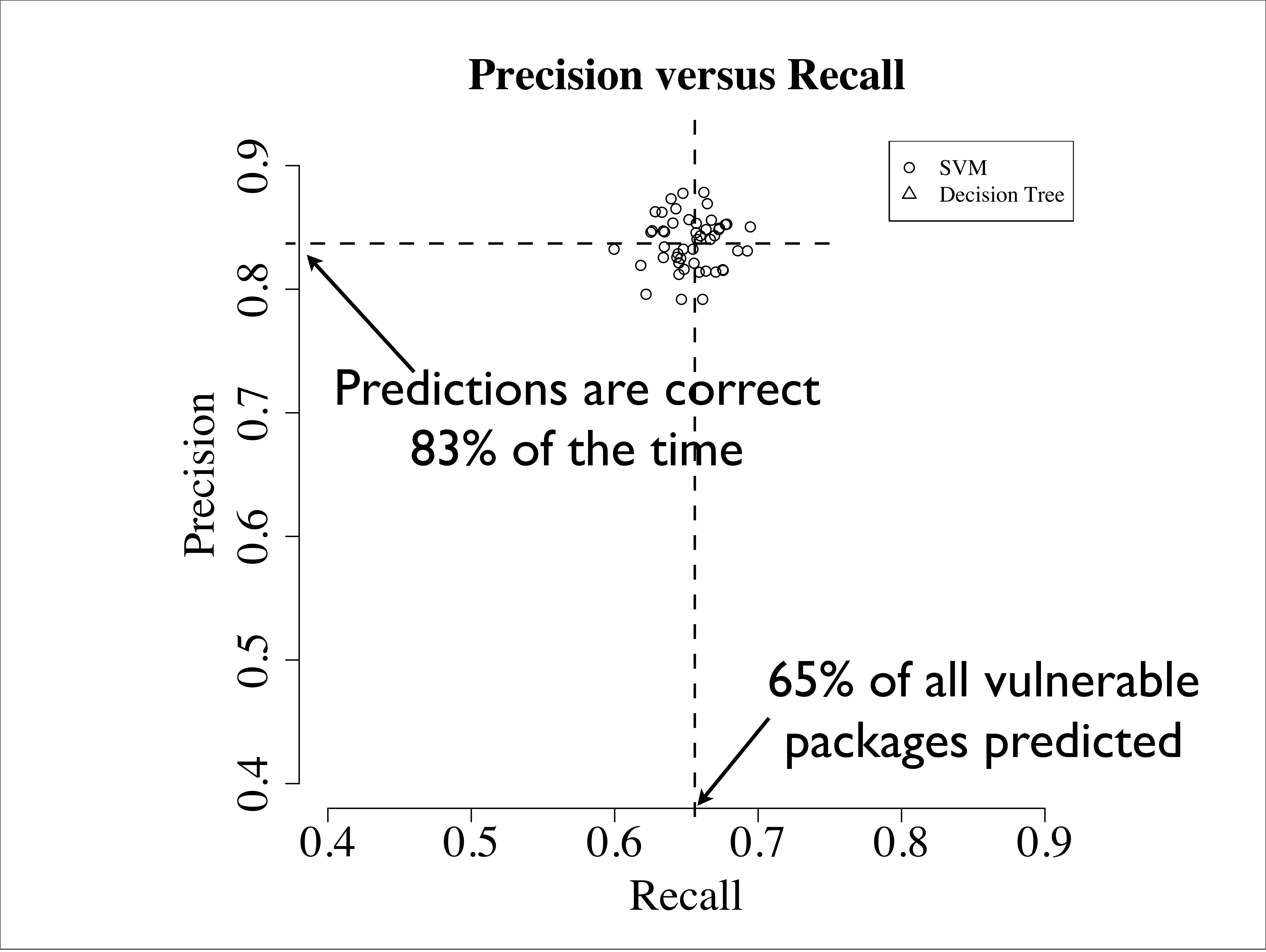
Results of 50 random splits: train with 2/3 of the packages, predict with the rest, record precision and recall.



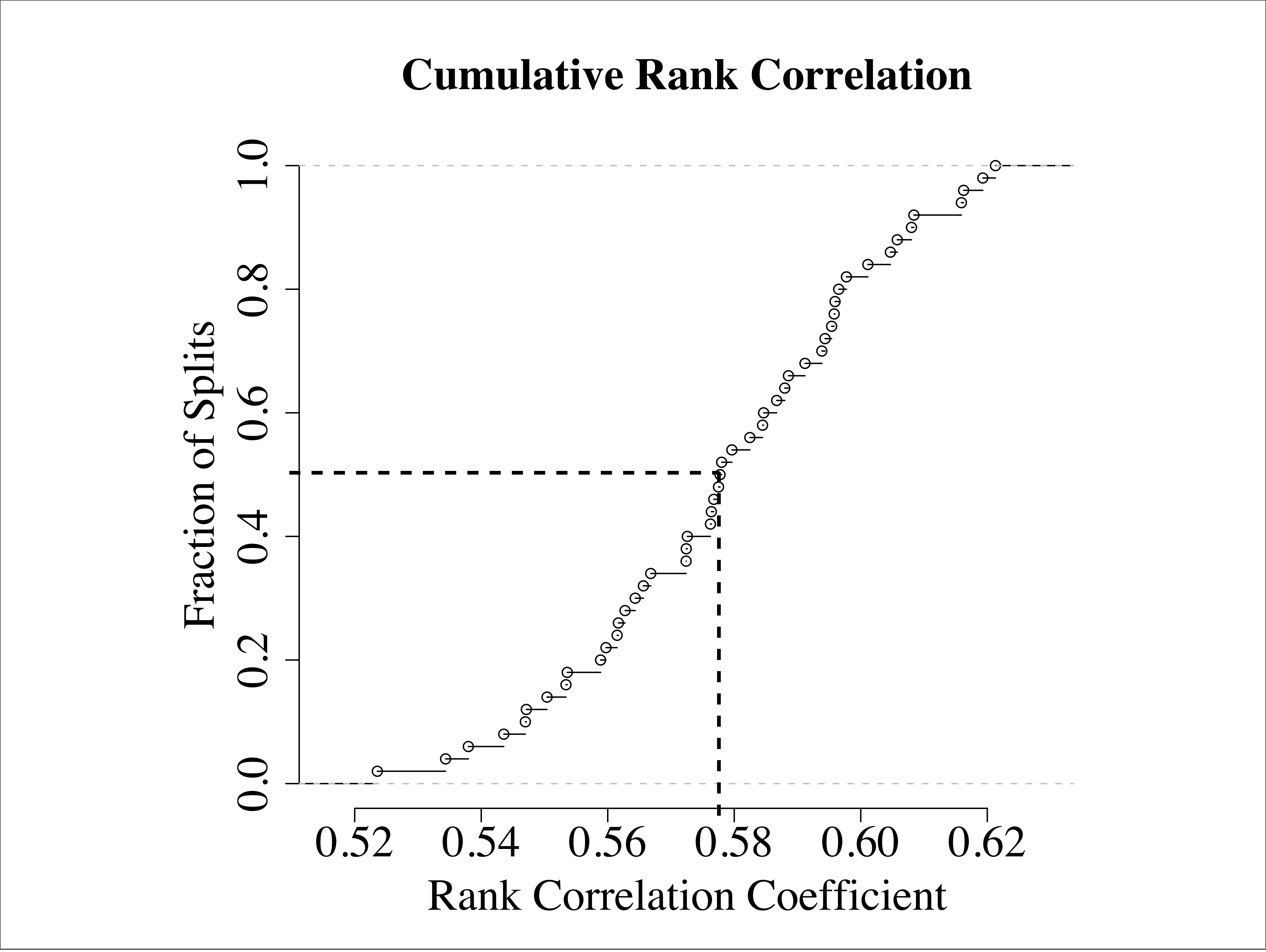
Results of 50 random splits: train with 2/3 of the packages, predict with the rest, record precision and recall.



Results of 50 random splits: train with 2/3 of the packages, predict with the rest, record precision and recall.



Results of 50 random splits: train with 2/3 of the packages, predict with the rest, record precision and recall.



Even though “self-evaluation” is a standard technique, what we realy want to know is if the method is able to predict the future... (next slide)

January 1, 2008

August 31, 2008

predict

Top 25 out of 2181

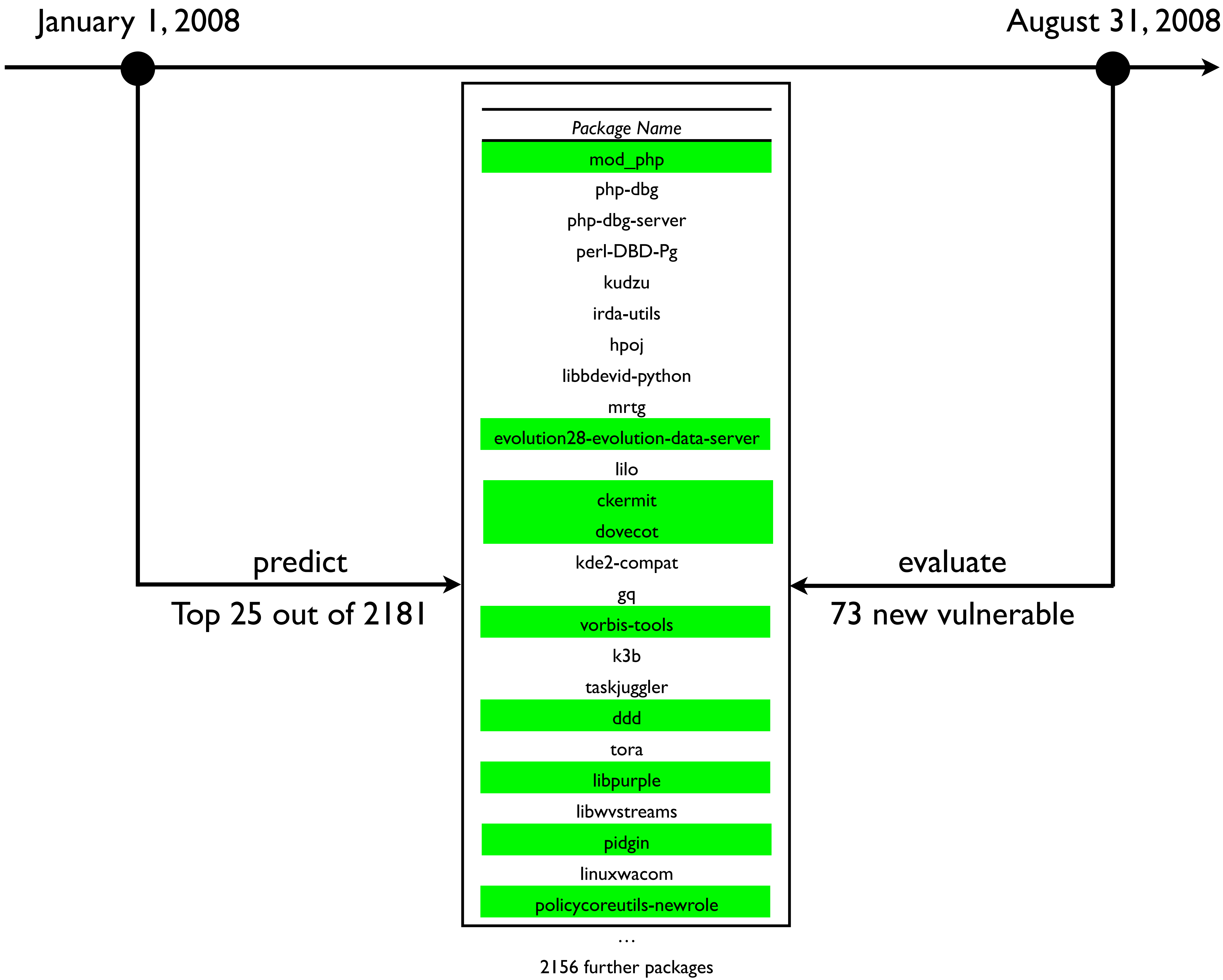
Package Name
mod_php
php-dbg
php-dbg-server
perl-DBD-Pg
kudzu
irda-utils
hpoj
libbdevid-python
mrtg
evolution28-evolution-data-server
lilo
ckermit
dovecot
kde2-compat
gq
vorbis-tools
k3b
taskjuggler
ddd
tora
libpurple
libwvstreams
pidgin
linuxwacom
policycoreutils-newrole

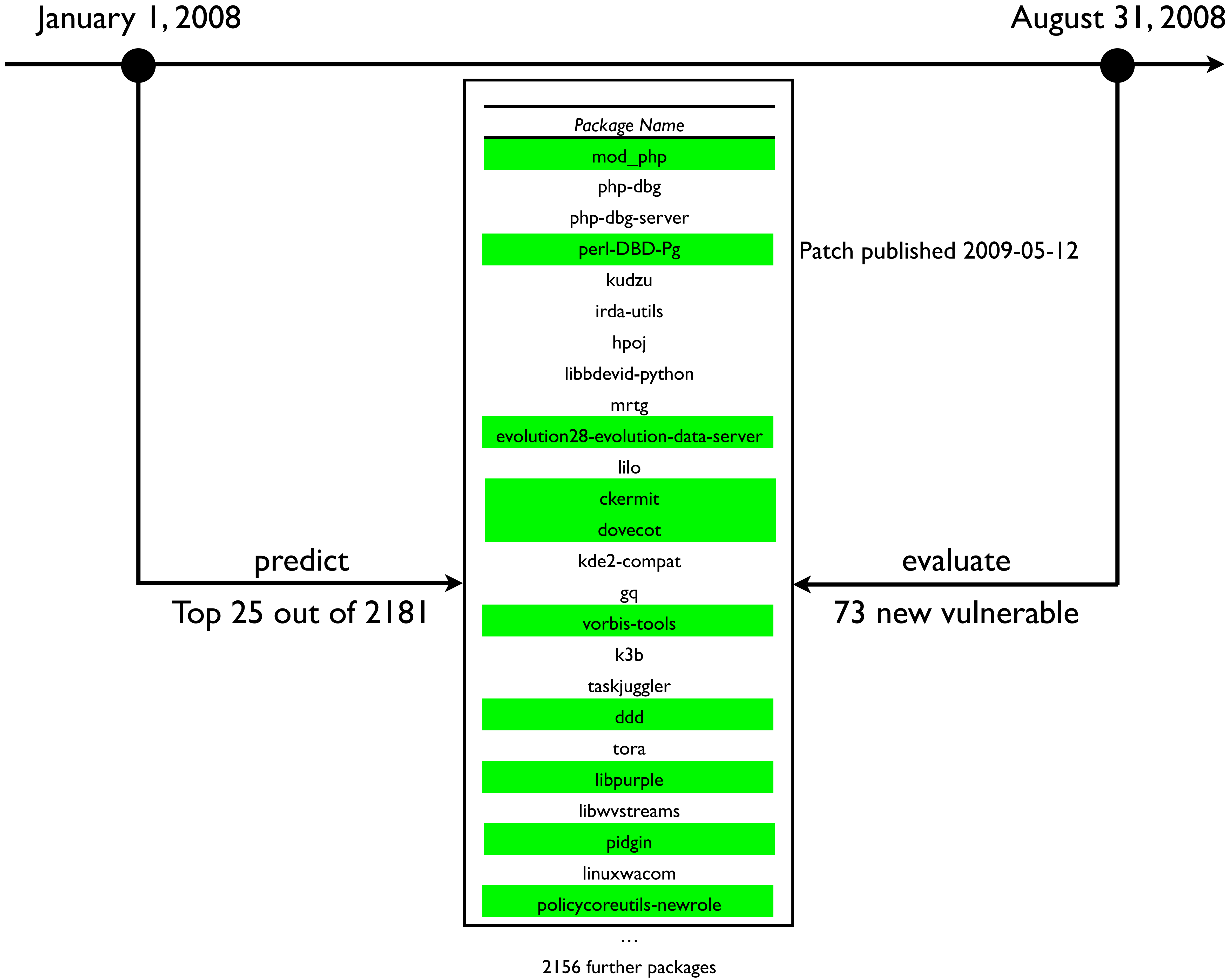
…

2156 further packages

evaluate

73 new vulnerable


January 1, 2008
August 31, 2008
predict
Top 25 out of 2181
evaluate
73 new vulnerable
Package Name
mod_php
php-dbg
php-dbg-server
perl-DBD-Pg
kudzu
irda-utils
hpoj
libbdevid-python
mrtg
evolution28-evolution-data-server
lilo
ckermit
dovecot
kde2-compat
gq
vorbis-tools
k3b
taskjuggler
ddd
tora
libpurple
libwvstreams
pidgin
linuxwacom
policycoreutils-newrole
…
2156 further packages



January 1, 2008
August 31, 2008
Package Name
mod_php
php-dbg
php-dbg-server
perl-DBD-Pg
Patch published 2009-05-12
kudzu
irda-utils
hpoj
libbdevid-python
mrtg
evolution28-evolution-data-server
lilo
ckermit
dovecot
kde2-compat
gq
vorbis-tools
k3b
taskjuggler
ddd
tora
libpurple
libwvstreams
pidgin
linuxwacom
policycoreutils-newrole
...
2156 further packages
predict
Top 25 out of 2181
evaluate
73 new vulnerable

Consequences

- When *building new applications*, choose less risky dependencies
  - use *GNU-SASL* instead of *cyrus-sasl*, Gnome instead of KDE
- When *maintaining existing applications*, prioritise resources
  - look at *krb5-libs*, not at *gkermit*

Conclusions

- Vulnerabilities correlate with dependencies
- Identification of risky dependencies
- Prediction with high precision, recall, correlation

http://research.microsoft.com/projects/esm/
http://www.artdecode.de/

* Have we worked with Red Hat: yes, have received positive feedback
* Usage Data: nonexistent
* Explain Correlation: See previous slide: domains