PACJAM: Securing Dependencies Continuously via Package-Oriented Debloating

Pardis Pashakhanloo, Aravind Machiry, Hyonyoung Choi, Anthony Canino, Kihong Heo, Insup Lee, and Mayur Naik.

MODERN SOFTWARE = ONE-SIZE-FITS-ALL

Package Managers

nuget

Chocolatey

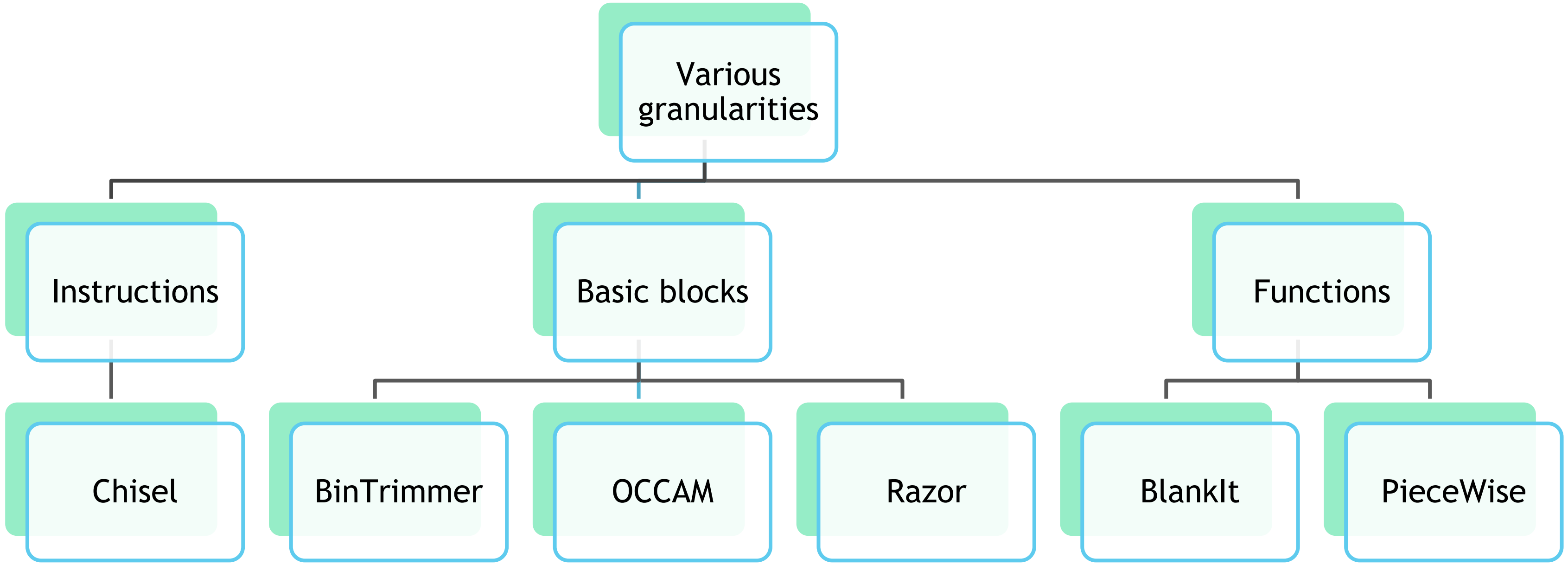
python™ PIP

npm

apt-get

SOFTWARE DEBLOATING

Removal of code artifacts that are not needed for any or certain use-cases.



PRACTICAL SOFTWARE DEBLOATING?

	Configurable Fallback Mechanism	Full Specs not Required	No Extra Runtime Requirements	Rapid Security Response
BinTrimmer	✗	✓	✓	✗
BlankIt	✓	✓	✗	✗
Chisel	✗	✗	✓	✗
Occam	✗	✗	✓	✗
PieceWise	✗	✓	✗	✗
Razor	✗	✗	✓	✗
PacJam	✓	✓	✓	✓

PACJAM:
ADDRESS SOFTWARE BLOAT AT THE PACKAGE-LEVEL

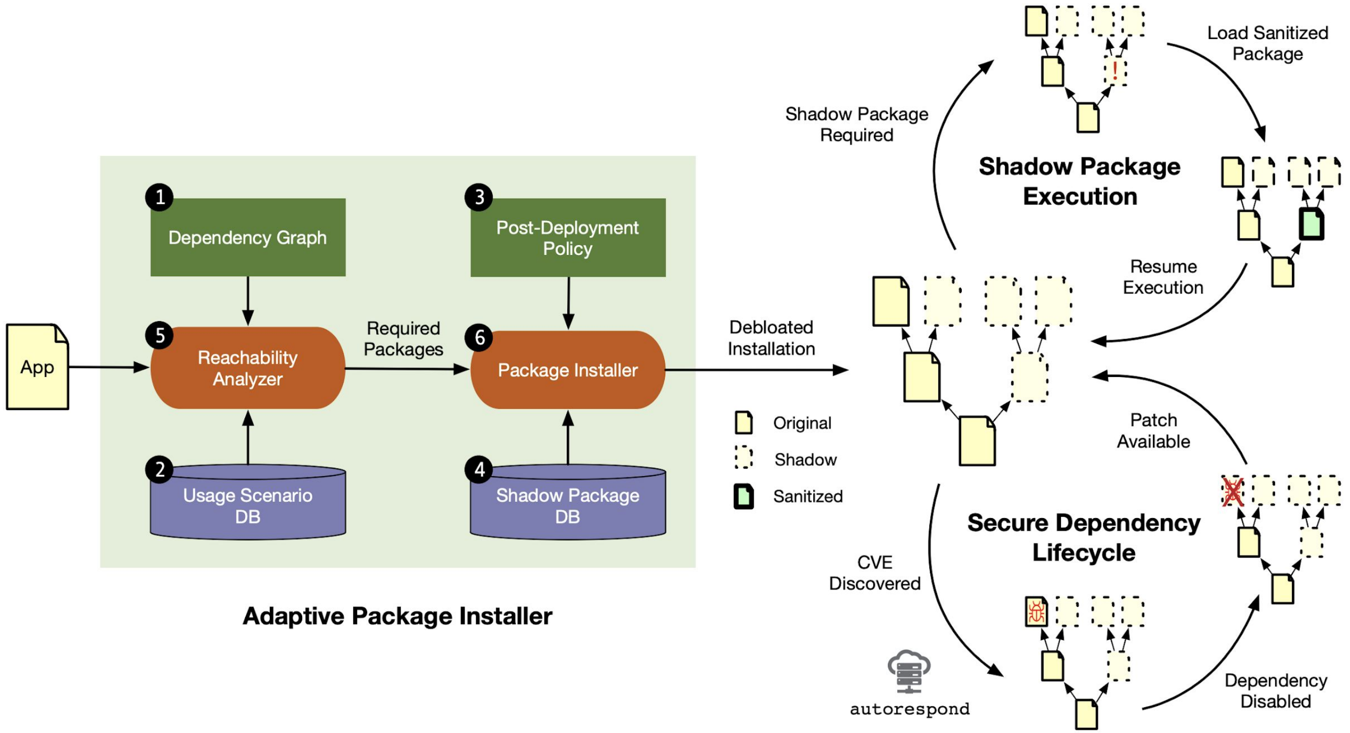
Debloating at the package level enables a generic debloating solution that is applicable to a wide range of applications.

PacJam is a package-level debloating framework that overcomes the mentioned limitations. PacJam,

- removes all statically unreachable packages (58% of packages per application),
- removes reachable but unused packages (8% of packages per application),
- and disables/enables packages on demand (shadow packages)

Example:
65% of Firefox’s dependent packages are not required!

OVERVIEW OF PACJAM



1 DEPENDENCY GRAPH

PacJam maintains the information of all dependencies between packages.

Direct dependencies of each package are available from the specified application metadata.

From this information, PacJam computes all indirect dependencies.

Example:
VLC 3.0.2 for Debian has 479 dependent packages, 10 of which are direct dependencies. 324 out of 479 are statically reachable.

2 USAGE SCENARIO DATABASE

PacJam captures application features by observing which dependent packages the application uses under each usage scenario.

We construct a usage scenario database for each application from:

- Common use cases
- Application tutorials
- Questions from Stack-Overflow
- Developer-provided test suite

Example:
For VLC, we collected 299 media files to include as many supported media formats as possible. Among the 155 dependent packages with shared libraries identified by the static analyzer, only 134 dependent packages with shared libraries are exercised to play all the collected media files.

3 SHADOW PACKAGE DATABASE

Shadow Package is a stripped-down version of an original package which does not provide any functionalities but a stub. The stub allows seamless execution of the application in cases where the code in the original package is required.

For each package, PacJam generates the shadow and sanitized versions along with the original version. This allows automating the secure dependency life-cycle.

PacJam creates gutless ABI-compatible mock libraries, which we call **shadow libraries** that belong in part to larger **shadow packages**.

Example:
PacJam initially installs 134 packages for VLC to support of the usage scenarios. For the remaining 187 packages, it instead installs the shadow version.

4 POST-DEPLOYMENT POLICY

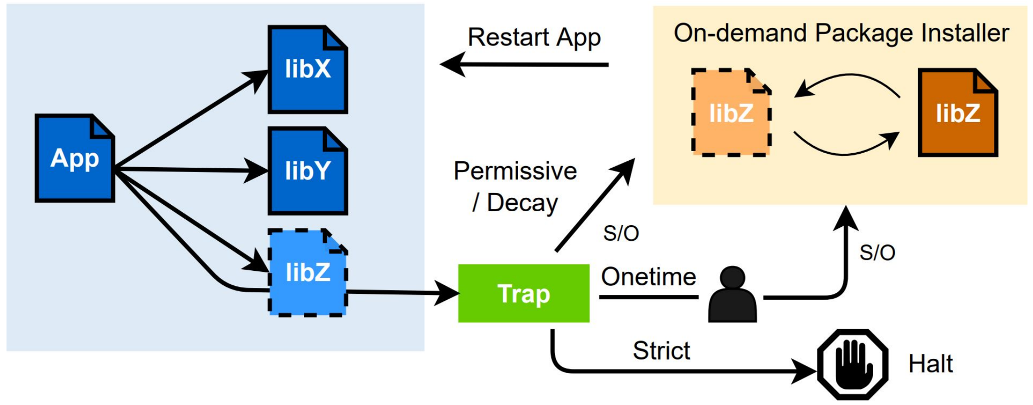
If an input requires the execution of a shadow package, our stub in the shadow package loads the sanitized version of the package and propagates execution to it.

This default behavior of installing sanitized packages can be changed by using various post-deployment policies.

Example 1:
A new vulnerability (CVE-2019-136159) is discovered. autorespond identifies that the corresponding vulnerable package is libebml and informs the package installer about libebml to disable it.

Example 2:
In VLC, if the sanitized version of libebml is used instead of the original libebml, the average runtime overhead over all the test cases is less than 200ms, and over the test cases specifically using this package is less than 1s.

4 POST-DEPLOYMENT POLICY



Shadow libraries allow customizing installations using PD policies that trap executions into removed functionality, called fault, and respond based on one of the following modes:

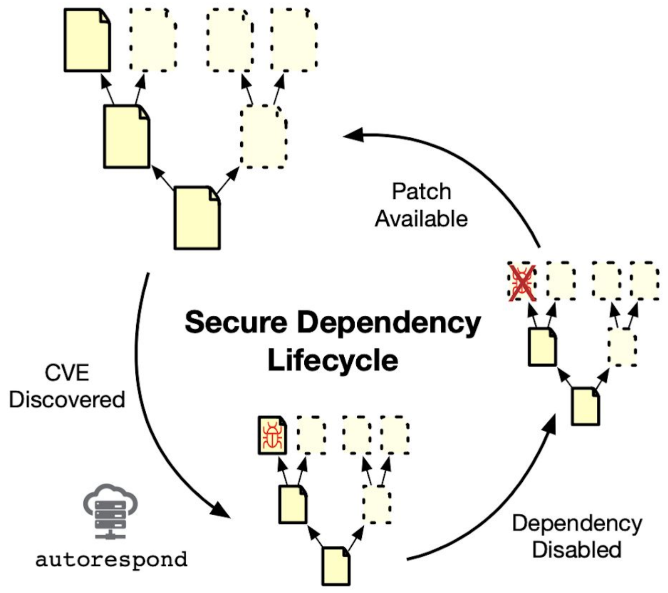
Strict Mode	Onetime Mode	Decay Mode	Permissive Mode

SECURE DEPENDENCY LIFECYCLE

We provide a suite of automation tools called *autorespond*, that maintains a secure dependency environment.

PacJam realizes continuous secure dependency lifecycle by integrating with existing vulnerability databases or discovering systems such as the CVE database, GitHub security alerts, or OSS-Fuzz.

PacJam bridges the gap between continuing to provide an application service with a known vulnerability and disabling the application until such a vulnerability is fixed.



EVALUATION

RQ1. Static Reachability

RQ2. Dynamic Reachability

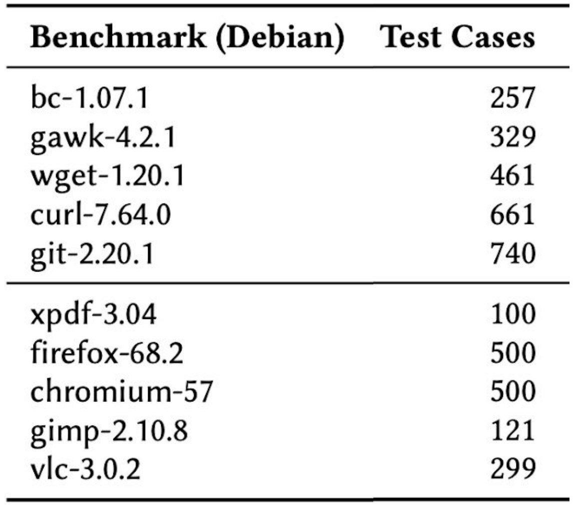
RQ3. Fallback Mechanism

RQ4. autorespond

EVALUATION METHODOLOGY

Benchmark Suite: 10 widely-used Linux applications.

Usage Scenarios



Benchmark (Debian)	Test Cases
bc-1.07.1	257
gawk-4.2.1	329
wget-1.20.1	461
curl-7.64.0	661
git-2.20.1	740
xpdf-3.04	100
firefox-68.2	500
chromium-57	500
gimp-2.10.8	121
vlc-3.0.2	299

Vulnerability Data

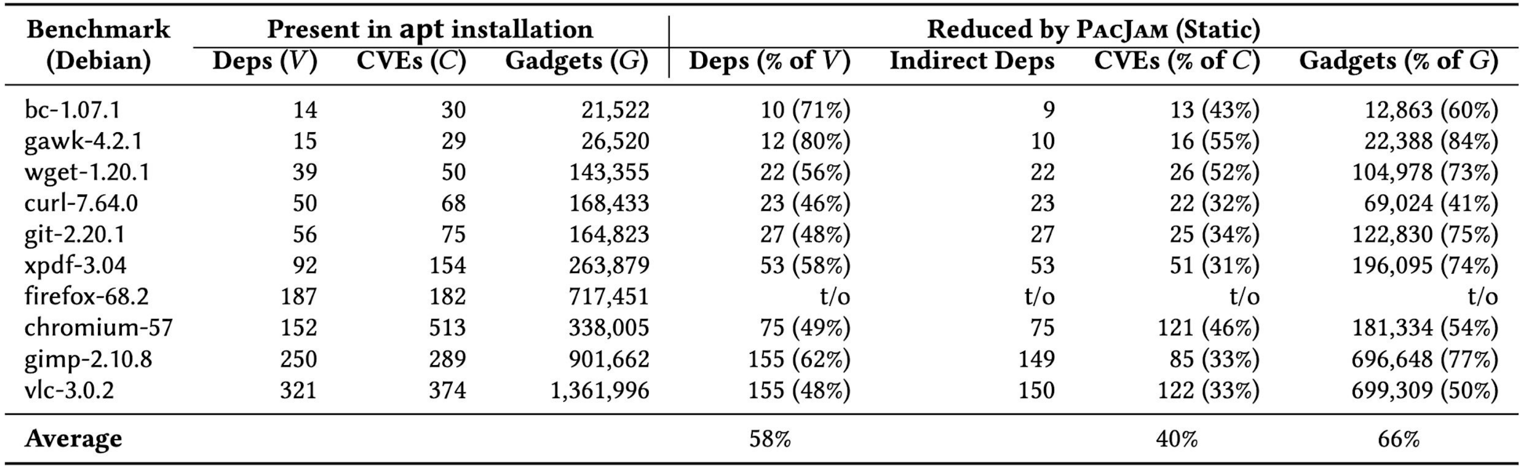
Information about each package's known vulnerabilities, i.e., CVEs collected from CVE databases.

Attack Surface

The number of CVEs and code reuse gadgets in its dependent packages.

[RQ1] EFFECTIVENESS OF STATIC REACHABILITY

We measure the effectiveness of PacJam at removing statically unreachable packages in terms of the number of removed dependencies, CVEs, and gadgets.

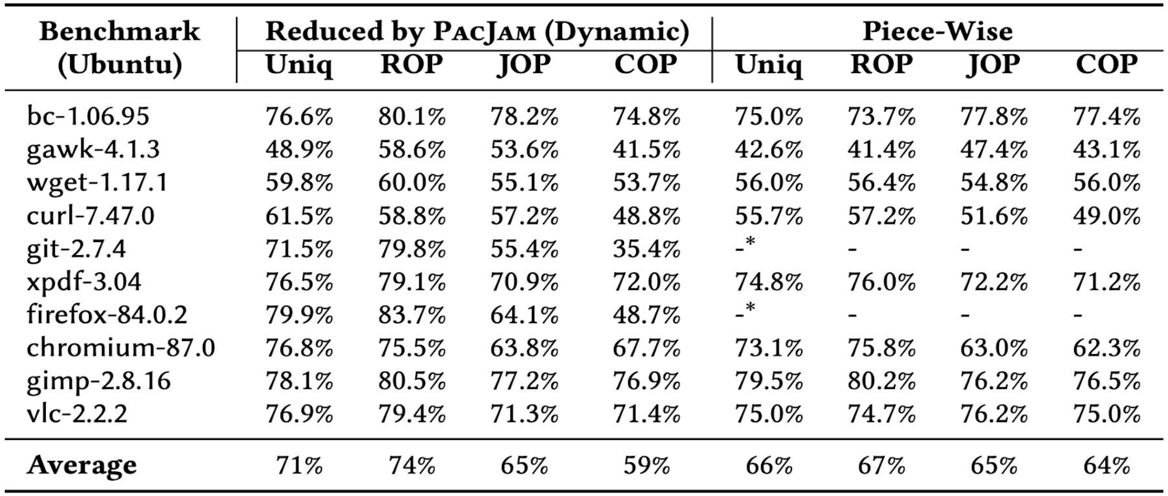


Benchmark (Debian)	**Present in apt installation**			**Reduced by PacJam (Static)**			
	Deps (V)	**CVEs (C)**	**Gadgets (G)**	**Deps (% of V)**	**Indirect Deps**	**CVEs (% of C)**	**Gadgets (% of G)**
bc-1.07.1	14	30	21,522	10 (71%)	9	13 (43%)	12,863 (60%)
gawk-4.2.1	15	29	26,520	12 (80%)	10	16 (55%)	22,388 (84%)
wget-1.20.1	39	50	143,355	22 (56%)	22	26 (52%)	104,978 (73%)
curl-7.64.0	50	68	168,433	23 (46%)	23	22 (32%)	69,024 (41%)
git-2.20.1	56	75	164,823	27 (48%)	27	25 (34%)	122,830 (75%)
xpdf-3.04	92	154	263,879	53 (58%)	53	51 (31%)	196,095 (74%)
firefox-68.2	187	182	717,451	t/o	t/o	t/o	t/o
chromium-57	152	513	338,005	75 (49%)	75	121 (46%)	181,334 (54%)
gimp-2.10.8	250	289	901,662	155 (62%)	149	85 (33%)	696,648 (77%)
vlc-3.0.2	321	374	1,361,996	155 (48%)	150	122 (33%)	699,309 (50%)
Average				58%		40%	66%

On average, PacJam removes 58% of packages across all the applications.

[RQ2] EFFECTIVENESS OF DYNAMIC REACHABILITY

PacJam can debloat applications further by removing dynamically unreachable packages for a given set of use cases.



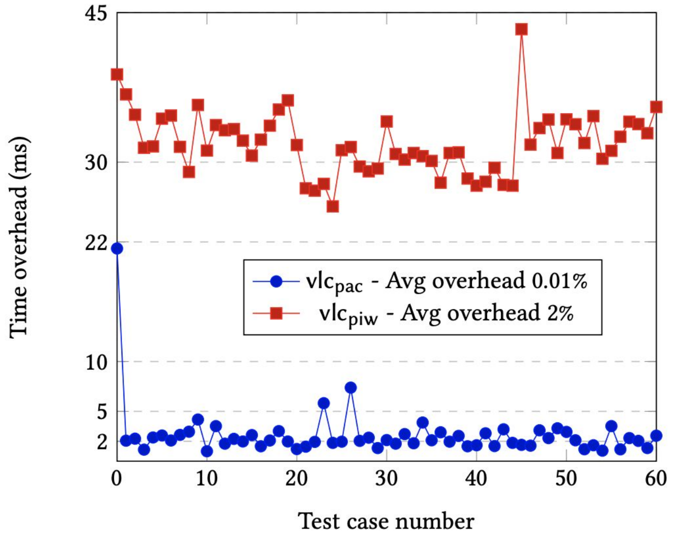
Benchmark (Ubuntu)	**Reduced by PacJam (Dynamic)**				**Piece-Wise**			
	Uniq	**ROP**	**JOP**	**COP**	**Uniq**	**ROP**	**JOP**	**COP**
bc-1.06.95	76.6%	80.1%	78.2%	74.8%	75.0%	73.7%	77.8%	77.4%
gawk-4.1.3	48.9%	58.6%	53.6%	41.5%	42.6%	41.4%	47.4%	43.1%
wget-1.17.1	59.8%	60.0%	55.1%	53.7%	56.0%	56.4%	54.8%	56.0%
curl-7.47.0	61.5%	58.8%	57.2%	48.8%	55.7%	57.2%	51.6%	49.0%
git-2.7.4	71.5%	79.8%	55.4%	35.4%	-*	-	-	-
xpdf-3.04	76.5%	79.1%	70.9%	72.0%	74.8%	76.0%	72.2%	71.2%
firefox-84.0.2	79.9%	83.7%	64.1%	48.7%	-*	-	-	-
chromium-87.0	76.8%	75.5%	63.8%	67.7%	73.1%	75.8%	63.0%	62.3%
gimp-2.8.16	78.1%	80.5%	77.2%	76.9%	79.5%	80.2%	76.2%	76.5%
vlc-2.2.2	76.9%	79.4%	71.3%	71.4%	75.0%	74.7%	76.2%	75.0%
Average	71%	74%	65%	59%	66%	67%	65%	64%

Debloating dynamically unreachable packages, removes an additional 8%.

~30% of vulnerabilities that PacJam prevents are of high severity.

[RQ3] EFFECTIVENESS OF FALLBACK MECHANISM

Due to incompleteness, static and dynamic unreachability analyses may fail to predict that some packages are required. Therefore, a fallback mechanism is required to deal with these cases.

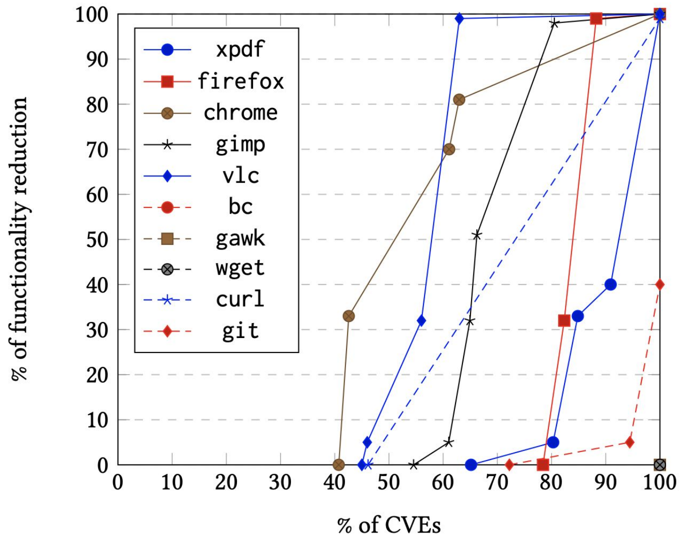


VLC_{piw} PieceWise-debloated VLC

VLC_{pac} PacJam-debloated VLC

[RQ4] EFFECTIVENESS OF AUTORESPOND

How effectively does autorespond attend to unpatched vulnerabilities?



Sometimes, vulnerable packages can be removed without compromising any common use case.

Example:
CVE-2017-5130 in libxml2.

Sometimes, disabling certain vulnerable packages affects 100% of an application's functionality.

Example:
CVE-2018-14550 in libpng.

RECAP

- We presented a package-oriented debloating framework, PacJam, for adaptive and security-aware management of an application’s dependent packages.
- PacJam enables package-level removal of security vulnerabilities in a manner that minimizes disruption to the application’s desired usage scenarios.
- Our experiments on a suite of 10 widely used Linux applications demonstrate that PacJam can effectively debloat applications and provide rapid response to newly discovered vulnerabilities in already installed packages.

Thank you!