Luci

Loader-based Dynamic Software Updates for Off-the-shelf Shared Objects

July 10, 2023

Bernhard Heinloth, Peter Wägemann, and Wolfgang Schröder-Preikschat

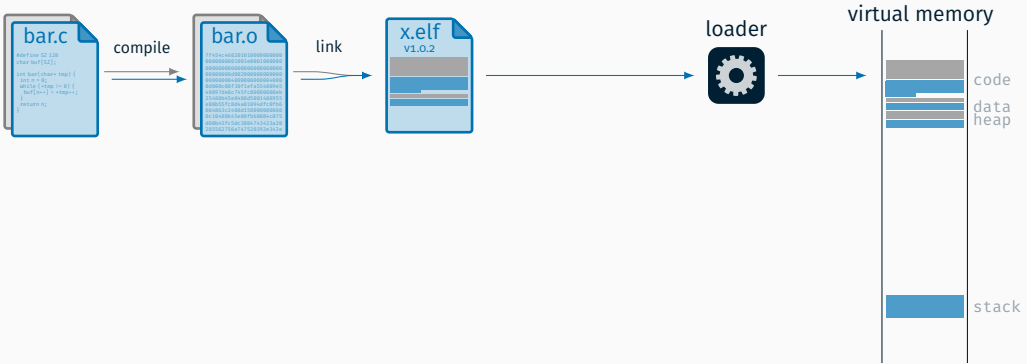
Friedrich-Alexander-Universität Erlangen-Nürnberg

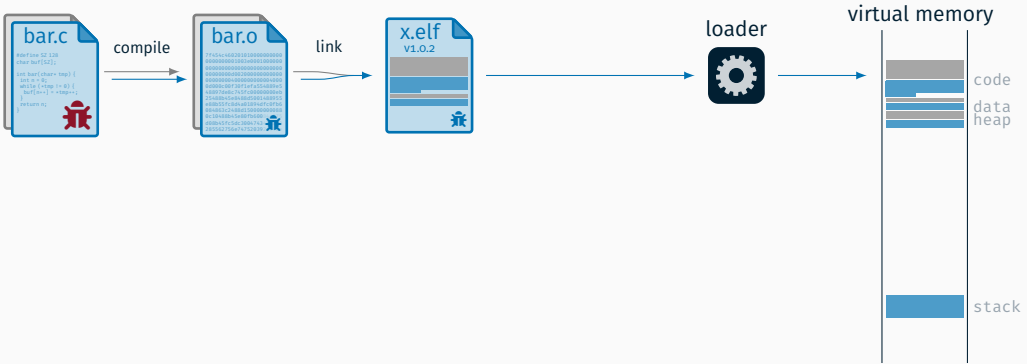
Friedrich-Alexander-Universität
Faculty of Engineering

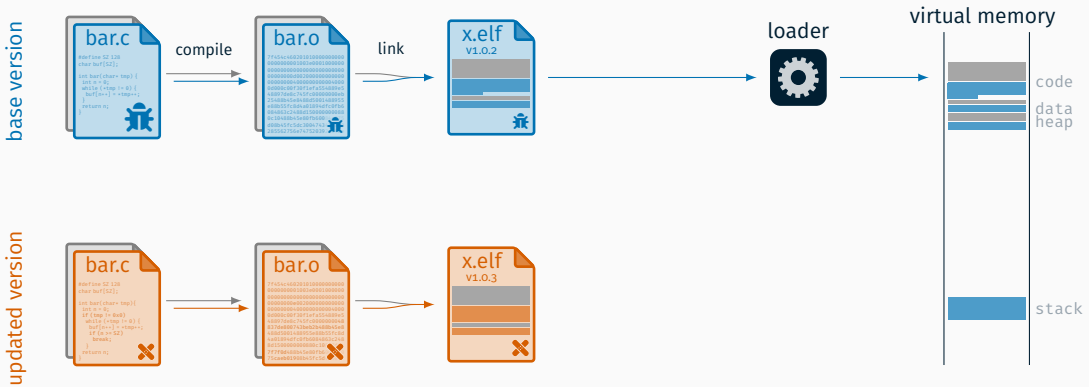
Chair in Distributed Systems
and Operating Systems


bar.c
#define SZ 128
char buf[SZ];
int bar(char* tmp) {
int n = 0;
while (*tmp != 0) {
buf[n++] = *tmp++;
}
return n;
}

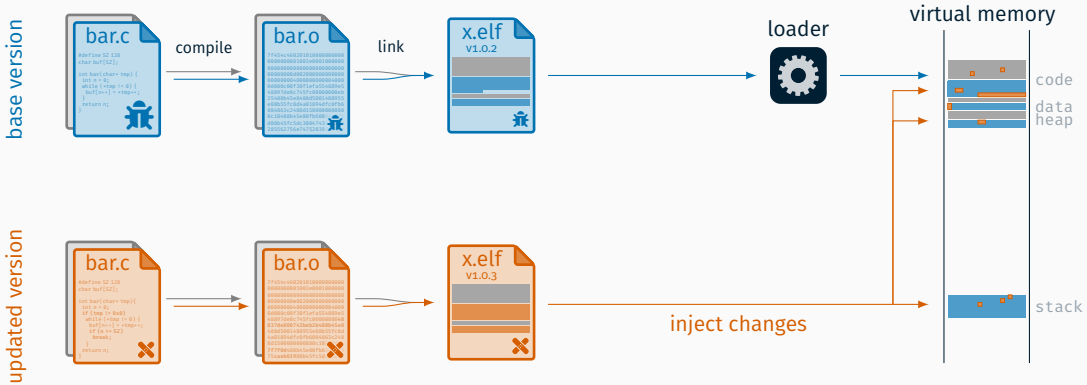

bar.c
compile
bar.o
link
x.elf
v1.0.2


bar.c
compile
bar.o
link
x.elf
v1.0.2
loader
virtual memory
code
data
heap
stack

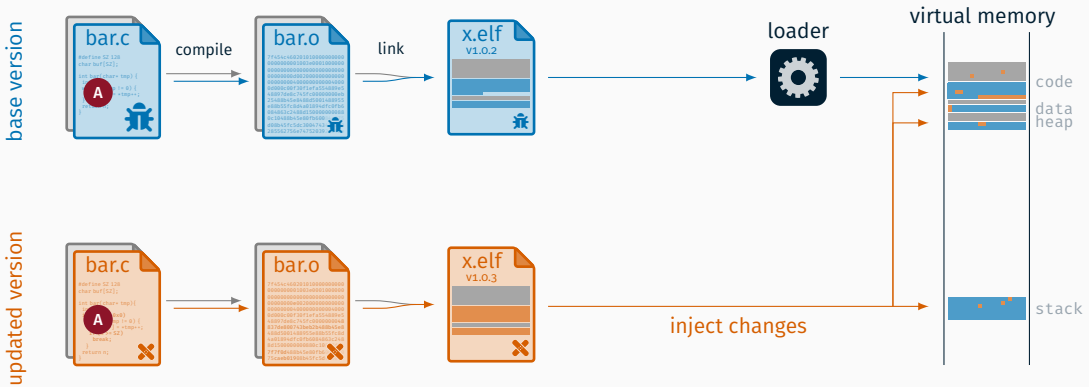

bar.c
compile
bar.o
link
x.elf
v1.0.2
loader
virtual memory
code
data
heap
stack


base version
bar.c
compile
bar.o
link
x.elf
v1.0.2
loader
virtual memory
code
data
heap
stack
updated version
bar.c
bar.o
x.elf
v1.0.3

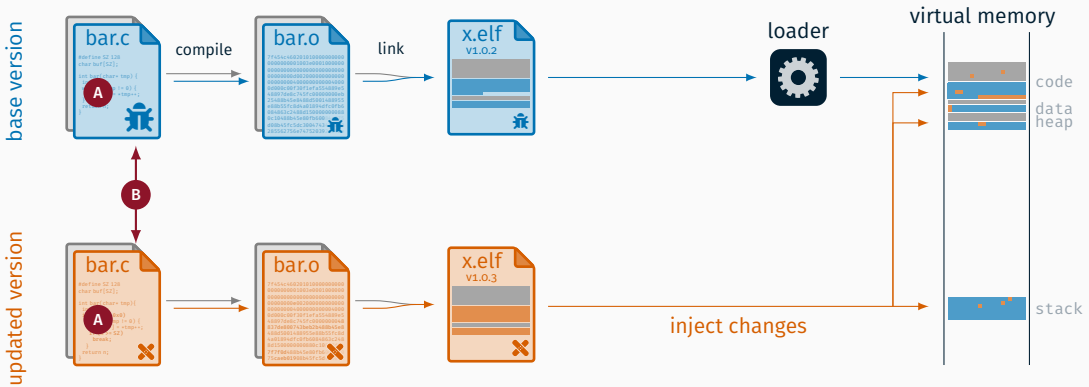
Dynamic Software Updates



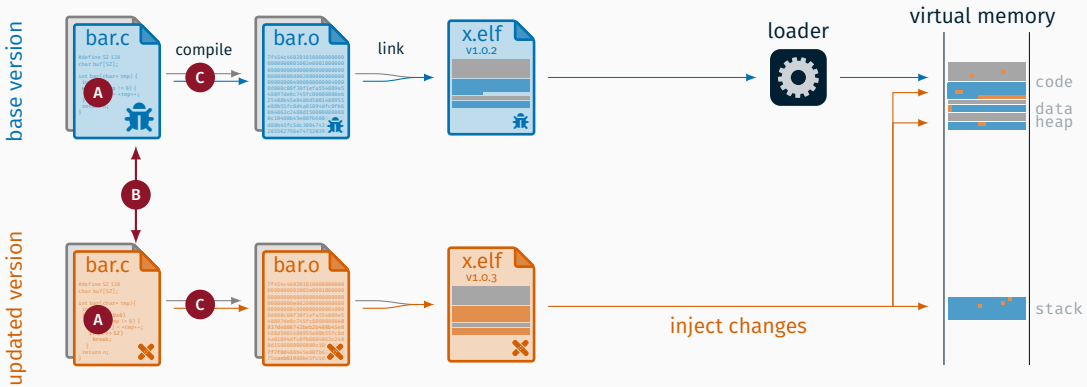
Dynamic Software Updates



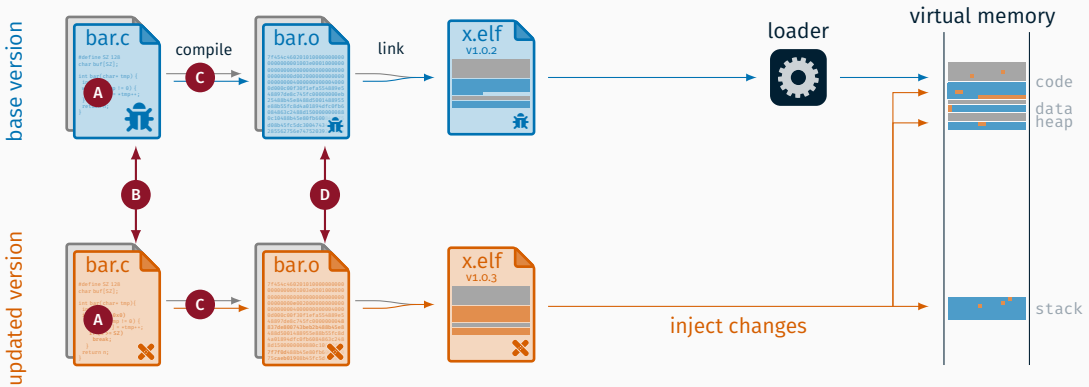
Dynamic Software Updates



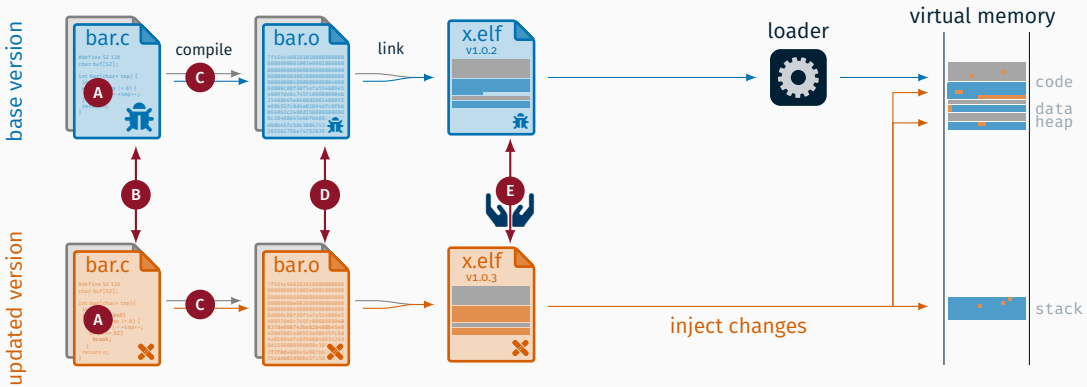
Dynamic Software Updates



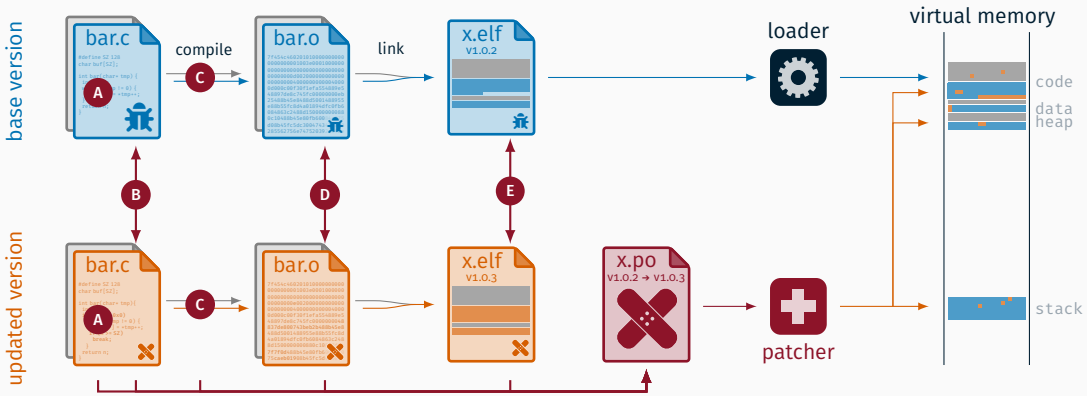
Dynamic Software Updates



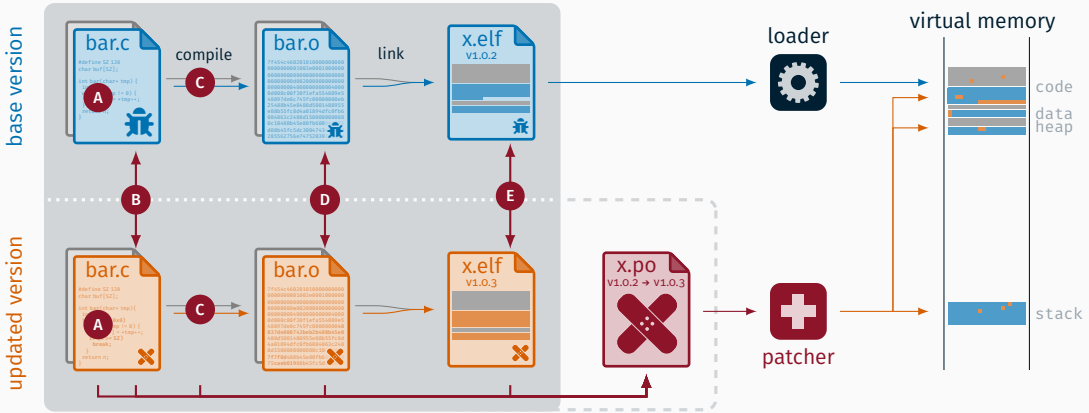
Dynamic Software Updates



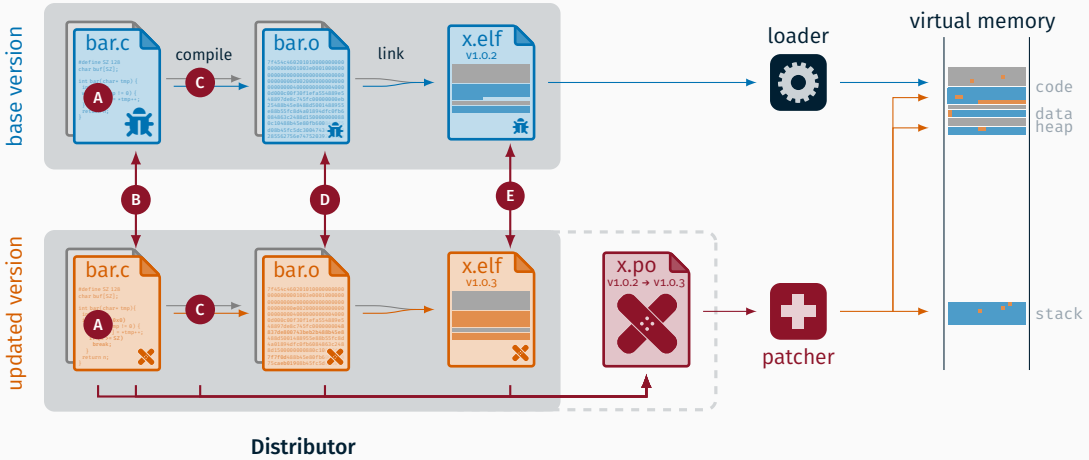
Dynamic Software Updates

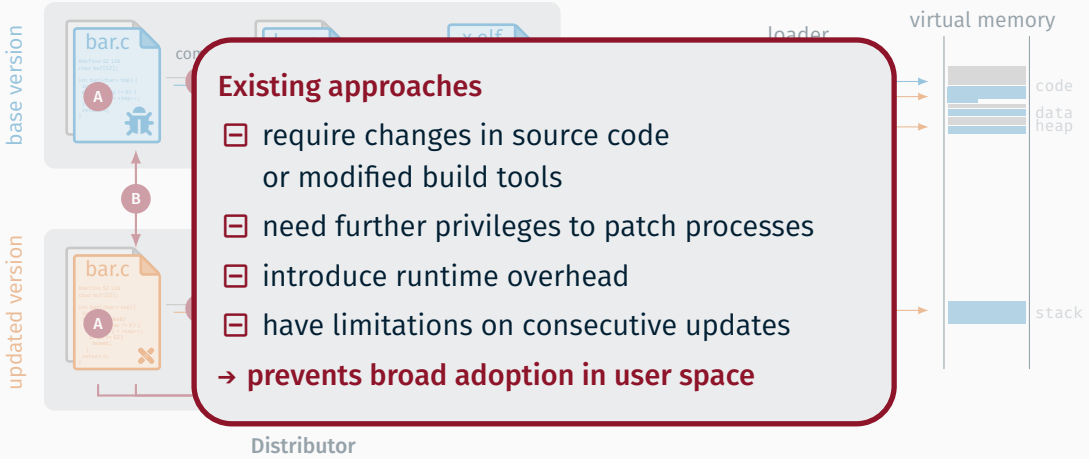


Dynamic Software Updates



Dynamic Software Updates



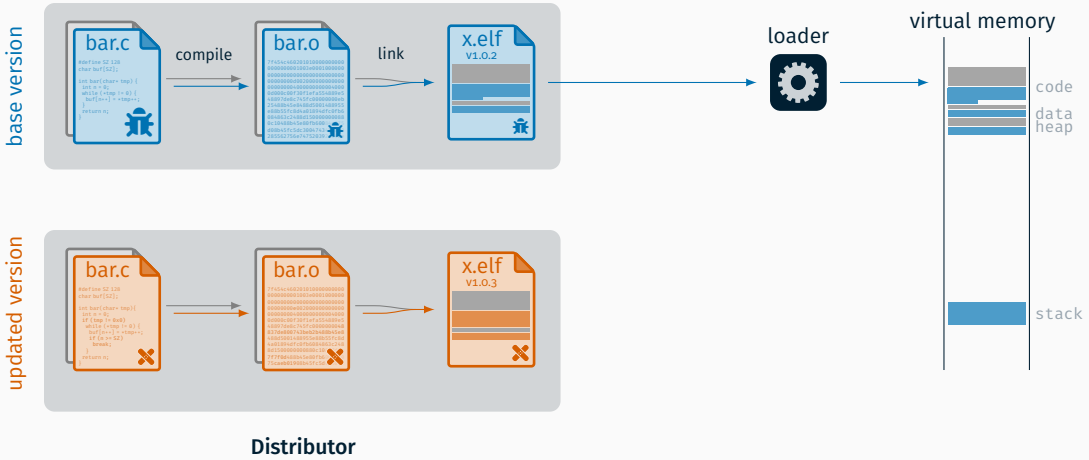


Existing approaches

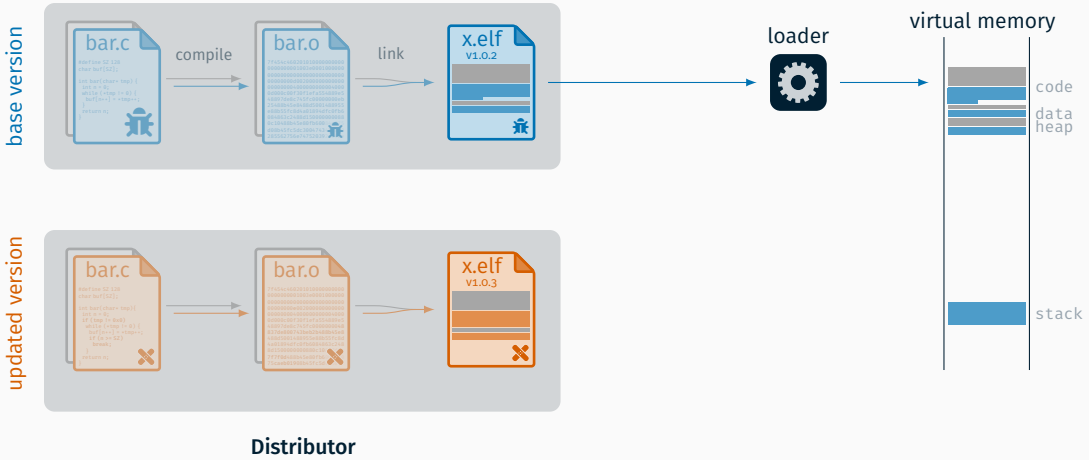
- ⊟ require changes in source code or modified build tools
- ⊟ need further privileges to patch processes
- ⊟ introduce runtime overhead
- ⊟ have limitations on consecutive updates

→ **prevents broad adoption in user space**

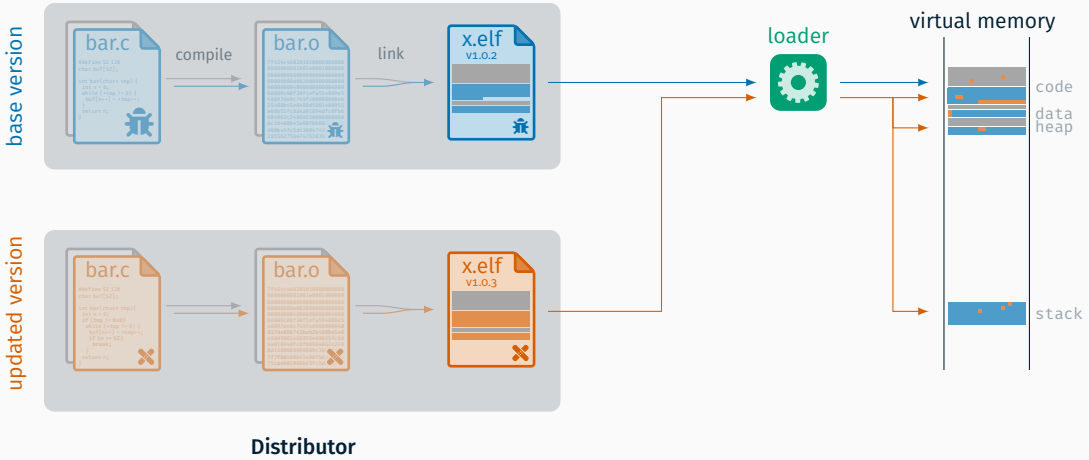
Dynamic Software Updates

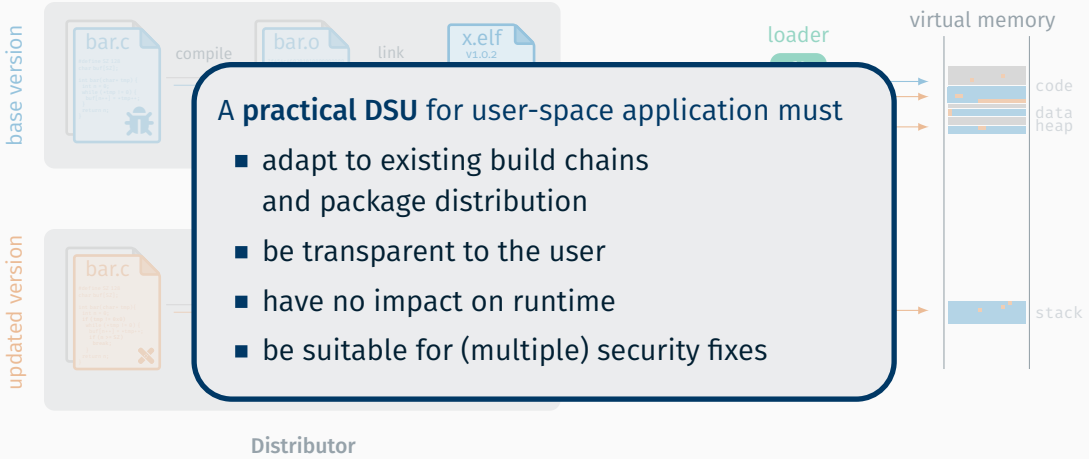


Dynamic Software Updates



Dynamic Software Updates





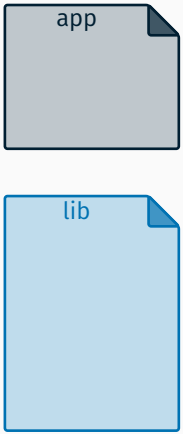
A **practical DSU** for user-space application must

- adapt to existing build chains and package distribution
- be transparent to the user
- have no impact on runtime
- be suitable for (multiple) security fixes

User-Space Applications

Application

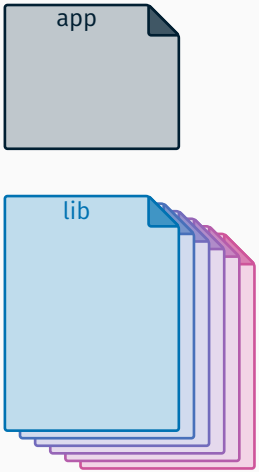
User-Space Applications



Application
using

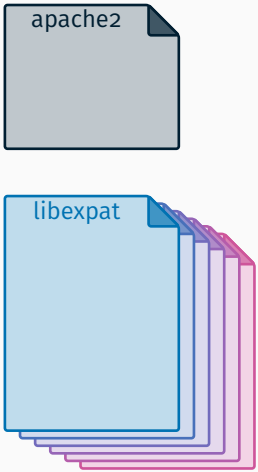
- standard library

User-Space Applications



Application
using

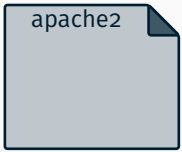
- standard library
- cryptography
- data (de-)compression
- parsing
- ...



Apache HTTP Server
using

- EXPAT
- GLIBC
- LIBXCRYPT
- OPENSSL
- PCRE
- ZLIB

(+ dynamically loaded modules and its dependencies)



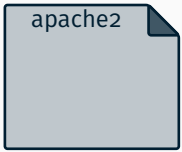
Apache HTTP Server
using

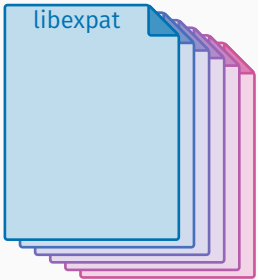
- EXPAT
- GLIBC
- LIBXCRYPT
- OPENSSL
- PCRE
- ZLIB

(+ dynamically loaded modules and its dependencies)

Fixed Expat Security Vulnerabilities in 2022

CVE ID	Impact	CVSS v2/v3	Fixed Release
CVE-2021-45960	denial of service	9.0 / 8.8	2.4.3 (Jan. 16, 2022)
CVE-2021-46143	denial of service	6.8 / 7.8	2.4.3
CVE-2022-22822	denial of service	7.5 / 9.8	2.4.3
CVE-2022-22823	denial of service	7.5 / 9.8	2.4.3
CVE-2022-22824	denial of service	7.5 / 9.8	2.4.3
CVE-2022-22825	denial of service	6.8 / 8.8	2.4.3
CVE-2022-22826	denial of service	6.8 / 8.8	2.4.3
CVE-2022-22827	denial of service	6.8 / 8.8	2.4.3
CVE-2022-23852	denial of service	7.5 / 9.8	2.4.4 (Jan. 30, 2022)
CVE-2022-23990	denial of service	5.0 / 7.5	2.4.4
CVE-2022-25235	code execution	7.5 / 9.8	2.4.5 (Feb. 18, 2022)
CVE-2022-25236	code execution	7.5 / 9.8	2.4.5 / 2.4.7 (March 4, 2022)
CVE-2022-25313	DoS [code exec?]	4.3 / 6.5	2.4.5 / 2.4.6 (Feb. 20, 2022)
CVE-2022-25314	denial of service	5.0 / 7.5	2.4.5
CVE-2022-25315	DoS [code exec?]	7.5 / 9.8	2.4.5
CVE-2022-40674	DoS [code exec?]	- / 8.1	2.4.9 (Sept. 20, 2022)
CVE-2022-43680	DoS [code exec?]	- / 7.5	2.5.0 (Oct. 25, 2022)



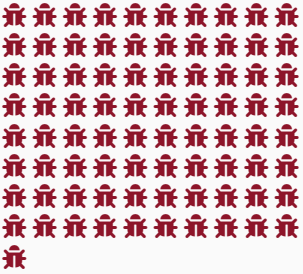


CVEs with CVSS v2 score ≥ 7.0

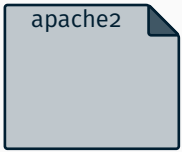
Apache HTTP Server

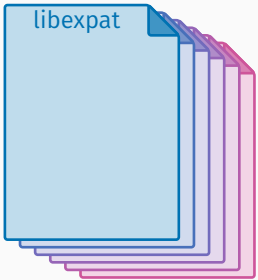
using

- EXPAT
- GLIBC
- LIBXCRYPT
- OPENSSL
- PCRE
- ZLIB



(+ dynamically loaded modules and its dependencies)

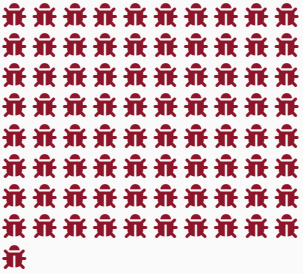




CVEs with CVSS v2 score ≥ 7.0

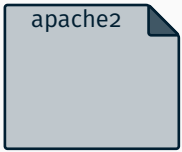
Apache HTTP Server
using

- EXPAT
- GLIBC
- LIBXCRYPT
- OPENSSL
- PCRE
- ZLIB



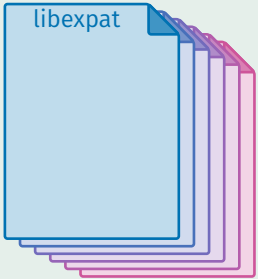
(+ dynamically loaded modules and its dependencies)

User-Space Applications



CVEs with CVSS v2 score ≥ 7.0

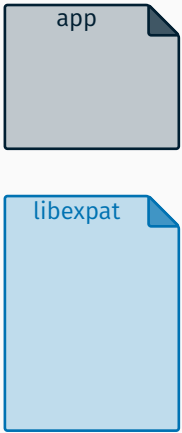
Apache HTTP Server
using

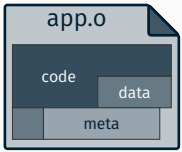


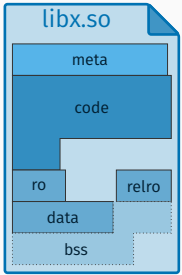
- Expat
- glibc
- libxcrypt
- OpenSSL
- PCRE
- Zlib

(+ dynamically loaded modules and its dependencies)

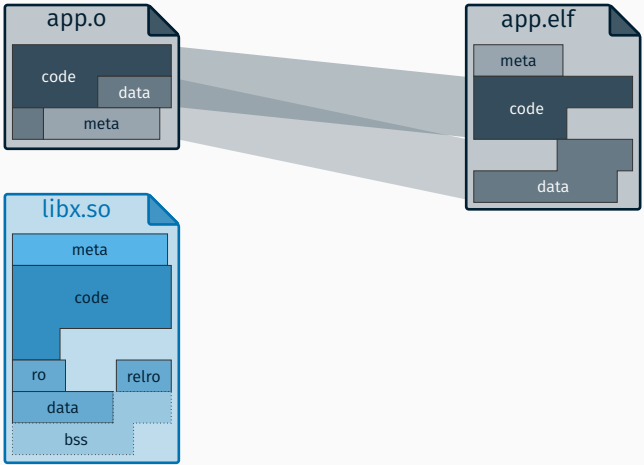
Library Linkage



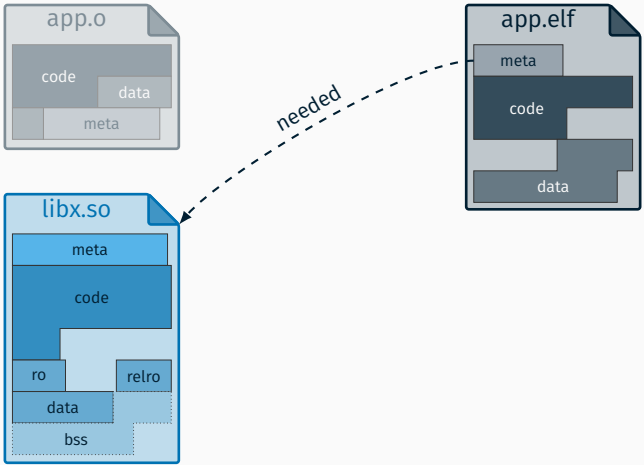

app.o
code
data
meta


libx.so
meta
code
ro
relro
data
bss

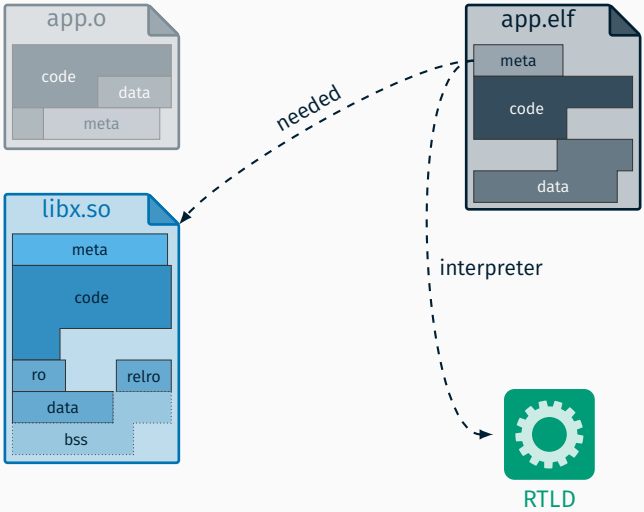
Library Linkage



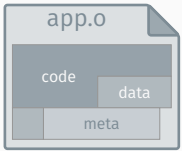
Library Linkage

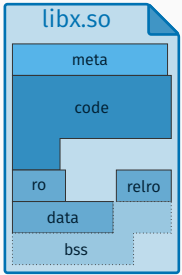


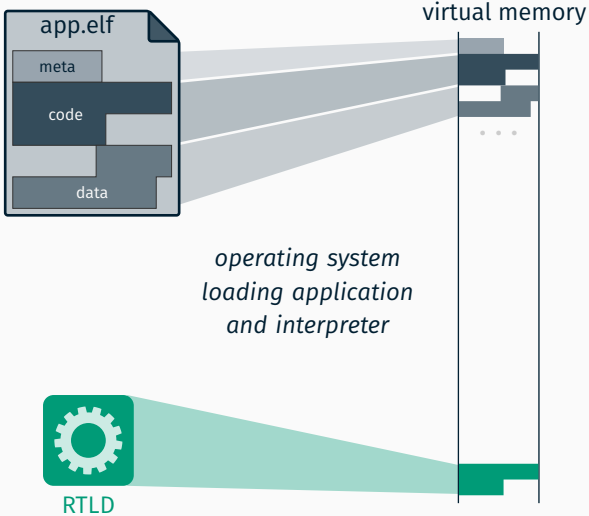
Library Linkage

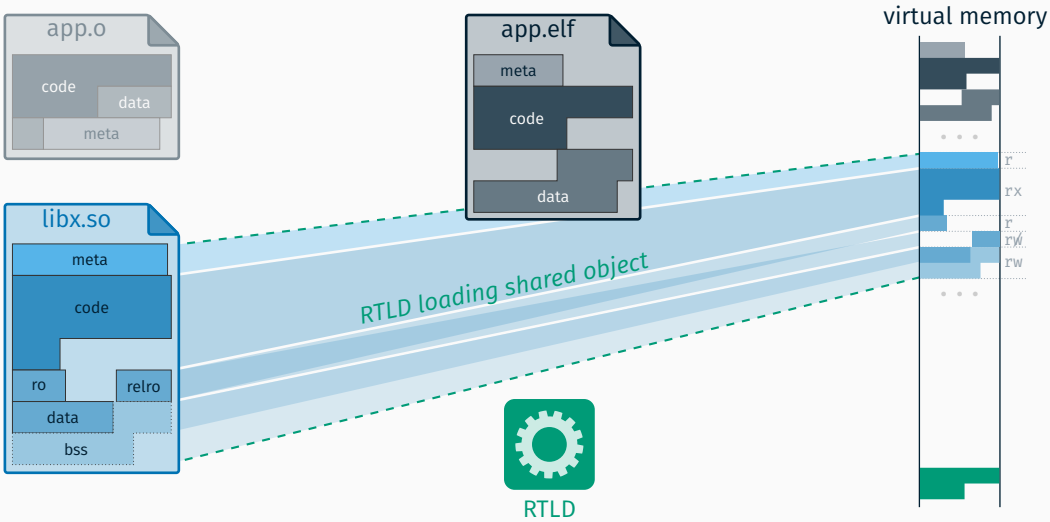


Library Linkage

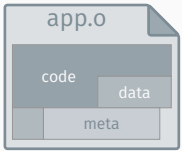


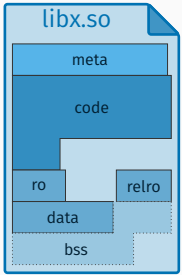


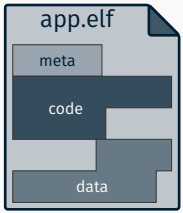



app.o
code
data
meta
app.elf
meta
code
data
virtual memory
libx.so
meta
code
ro
relro
data
bss
RTLD loading shared object
r
rx
r
rw
rw
RTLD

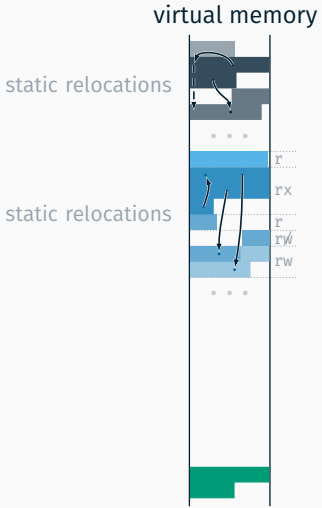
Library Linkage



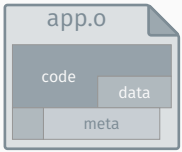


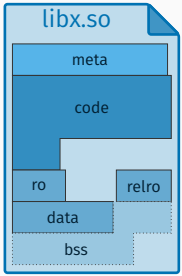


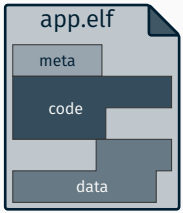
RTLD



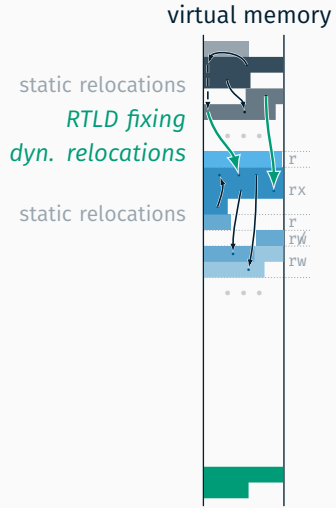
Library Linkage







RTLD



Approach

- RTLD has meta information and access to process virtual memory

Luci

Luci Approach

- RTLD has meta information and access to process virtual memory
- library functions eventually *return*

Luci

- RTLD has meta information and access to process virtual memory
- library functions eventually *return*
- no structural changes in security fixes
  - same interface (API)
  - only code changes

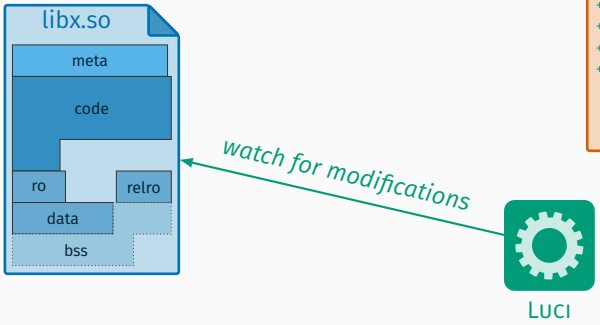
CVE-2022-23852.patch

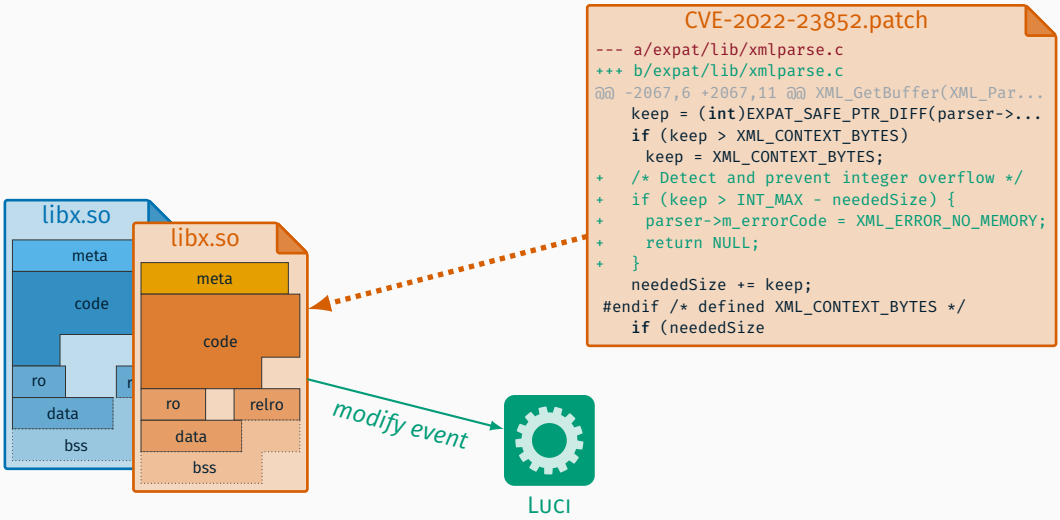
```
--- a/expat/lib/xmlparse.c
+++ b/expat/lib/xmlparse.c
@@ -2067,6 +2067,11 @@ XML_GetBuffer(XML_Par...
    keep = (int)EXPAT_SAFE_PTR_DIFF(parser->...
    if (keep > XML_CONTEXT_BYTES)
      keep = XML_CONTEXT_BYTES;
+   /* Detect and prevent integer overflow */
+   if (keep > INT_MAX - neededSize) {
+     parser->m_errorCode = XML_ERROR_NO_MEMORY;
+     return NULL;
+   }
    neededSize += keep;
 #endif /* defined XML_CONTEXT_BYTES */
    if (neededSize
```

Luci

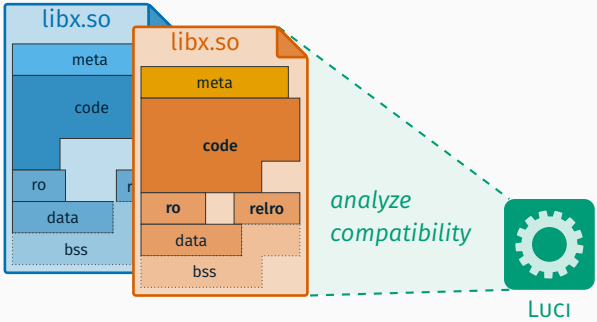
CVE-2022-23852.patch

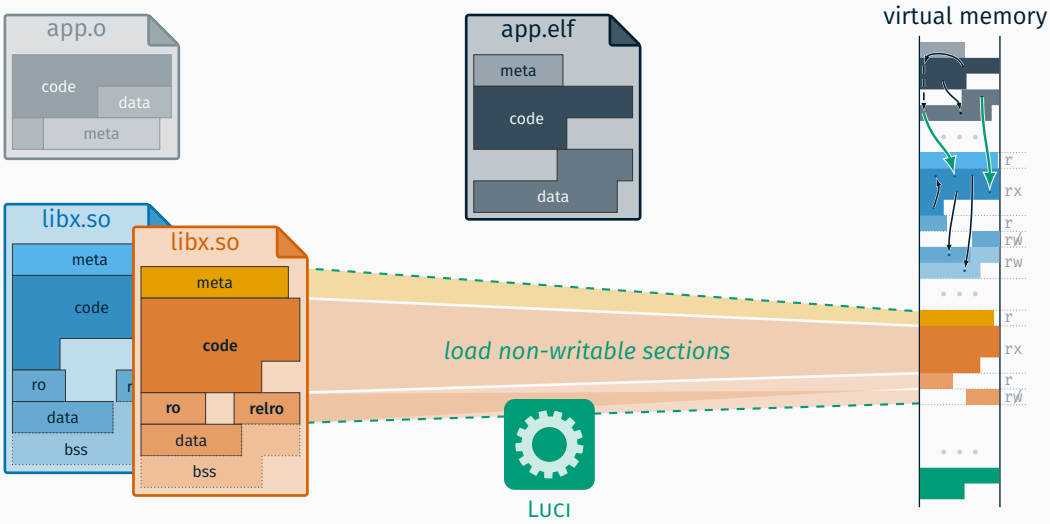
```
--- a/expat/lib/xmlparse.c
+++ b/expat/lib/xmlparse.c
@@ -2067,6 +2067,11 @@ XML_GetBuffer(XML_Par...
    keep = (int)EXPAT_SAFE_PTR_DIFF(parser->...
    if (keep > XML_CONTEXT_BYTES)
      keep = XML_CONTEXT_BYTES;
+   /* Detect and prevent integer overflow */
+   if (keep > INT_MAX - neededSize) {
+     parser->m_errorCode = XML_ERROR_NO_MEMORY;
+     return NULL;
+   }
    neededSize += keep;
 #endif /* defined XML_CONTEXT_BYTES */
    if (neededSize
```

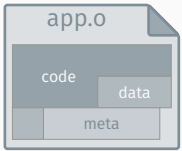


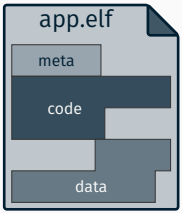

CVE-2022-23852.patch
--- a/expat/lib/xmlparse.c
+++ b/expat/lib/xmlparse.c
@@ -2067,6 +2067,11 @@ XML_GetBuffer(XML_Par...
keep = (int)EXPAT_SAFE_PTR_DIFF(parser->...
if (keep > XML_CONTEXT_BYTES)
keep = XML_CONTEXT_BYTES;
+ /* Detect and prevent integer overflow */
+ if (keep > INT_MAX - neededSize) {
+ parser->m_errorCode = XML_ERROR_NO_MEMORY;
+ return NULL;
+ }
neededSize += keep;
#endif /* defined XML_CONTEXT_BYTES */
if (neededSize
libx.so
meta
code
ro
data
bss
libx.so
meta
code
ro
relro
data
bss
modify event
Luci

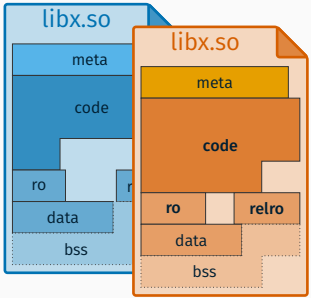
Luci Approach



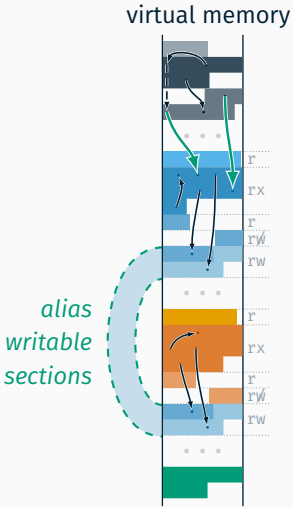

app.o
code
data
meta
app.elf
meta
code
data
virtual memory
libx.so
meta
code
ro
data
bss
libx.so
meta
code
ro
relro
data
bss
load non-writable sections
r
rx
r
rw
rw
r
rx
r
rw
Luci

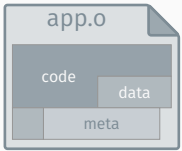

app.o
code
data
meta

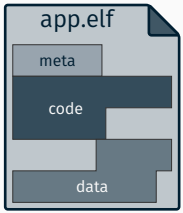

app.elf
meta
code
data

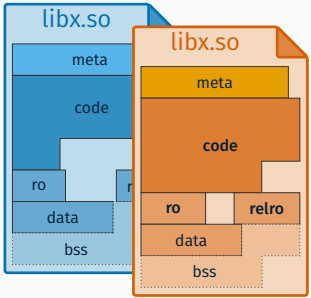

libx.so
meta
code
ro
data
bss
libx.so
meta
code
ro
relro
data
bss

Luci

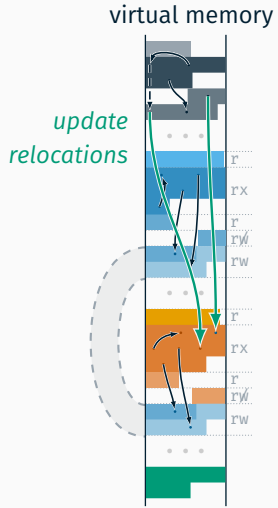

virtual memory
r
rx
r
rw
rw
r
rx
r
rw
rw
alias
writable
sections

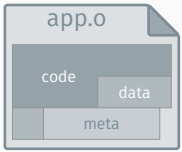

app.o
code
data
meta

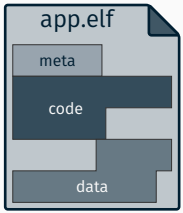

app.elf
meta
code
data

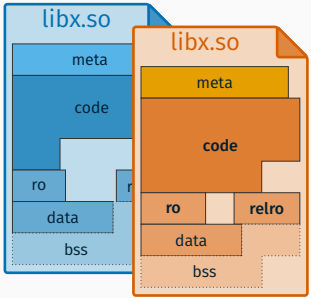

libx.so
meta
code
ro
r
data
bss
libx.so
meta
code
ro
relro
data
bss

Luci

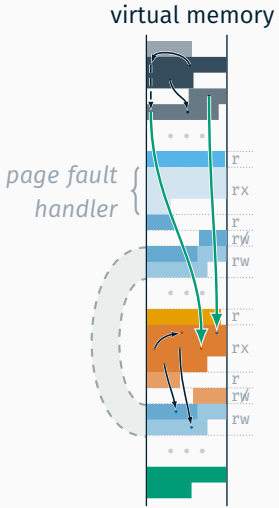

virtual memory
update
relocations
r
rx
r
rw
rw
r
rx
r
rw
rw


app.o
code
data
meta


app.elf
meta
code
data


libx.so
meta
code
ro
r
data
bss
libx.so
meta
code
ro
relro
data
bss


Luci


virtual memory
page fault
handler
r
rx
r
rw
rw
r
rx
r
rw
rw

Advantages of the Luci Approach

Advantages of the Luci Approach

- ⊞ comparatively simple and safe

Advantages of the Luci Approach

- ⊞ comparatively simple and safe
- ⊞ no quiescence required

Advantages of the Luci Approach

- ⊞ comparatively simple and safe
- ⊞ no quiescence required
- ⊞ agnostic to programming language and build tools

Advantages of the Luci Approach

- ⊞ comparatively simple and safe
- ⊞ no quiescence required
- ⊞ agnostic to programming language and build tools
- ⊞ no overhead during runtime

Advantages of the Luci Approach

- ⊞ comparatively simple and safe
- ⊞ no quiescence required
- ⊞ agnostic to programming language and build tools
- ⊞ no overhead during runtime
- ⊞ not requiring any additional permissions

Advantages of the Luci Approach

- ⊞ comparatively simple and safe
- ⊞ no quiescence required
- ⊞ agnostic to programming language and build tools
- ⊞ no overhead during runtime
- ⊞ not requiring any additional permissions
- ⊞ arbitrary update sequence

Implementation of Luci

- dynamic linker/loader (RTLD)
- basic *glibc* compatibility
- tracks file modifications of loaded shared objects

Implementation of Luci

- dynamic linker/loader (RTLD)
- basic *glibc* compatibility
- tracks file modifications of loaded shared objects
  - automatic compatibility check:
    - identical writable segment
    - unchanged initialization routines
    - no structural changes

Implementation of Luci

- dynamic linker/loader (RTLD)
- basic *glibc* compatibility
- tracks file modifications of loaded shared objects
  - automatic compatibility check:
    - identical writable segment → ELF / DWARF
    - unchanged initialization routines → ELF
    - no structural changes → ELF / DWARF

Implementation of Luci

- dynamic linker/loader (RTLD)
- basic *glibc* compatibility
- tracks file modifications of loaded shared objects
  - automatic compatibility check:
    - identical writable segment → ELF / DWARF
    - unchanged initialization routines → ELF
    - no structural changes → ELF / DWARF
  - on compatible library:
    1. loading to an unused memory area in process
    2. applying relocations (e.g., `.data.rel.ro`)
    3. updating all *Global Offset Table* entries referring to the changed library
    4. disabling old code segment (with some delay)

Implementation of Luci

- dynamic linker/loader (RTLD)
- basic *glibc* compatibility
- tracks file modifications of loaded shared objects
  - automatic compatibility check:
    - identical writable segment → ELF / DWARF
    - unchanged initialization routines → ELF
    - no structural changes → ELF / DWARF
  - on compatible library:
    1. loading to an unused memory area in process
    2. applying relocations (e.g., `.data.rel.ro`)
    3. updating all *Global Offset Table* entries referring to the changed library
    4. disabling old code segment (with some delay)
  - for incompatible library:
    1. notify user (IPC)
    2. resume with old version

Evaluation

Evaluation

Expat XML parser Version 2

- widely used
  - e.g., Chromium, Cmake, GDB, Git, Firefox, LibreOffice, PHP, Python, ...
- 27 releases (since 2006)
- 29 CVEs (since 2012)

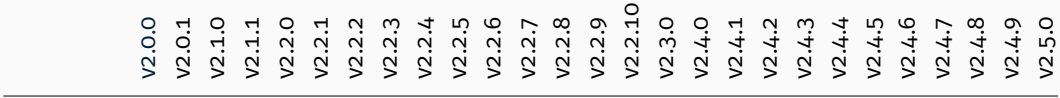
Evaluation

EXPAT XML PARSER Version 2

- widely used
  - e.g., Chromium, Cmake, GDB, Git, Firefox, LibreOffice, PHP, Python, ...
- 27 releases (since 2006)
- 29 CVEs (since 2012)

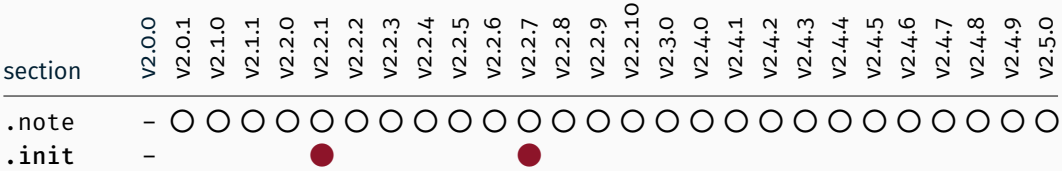
→ *vanilla* build of each release

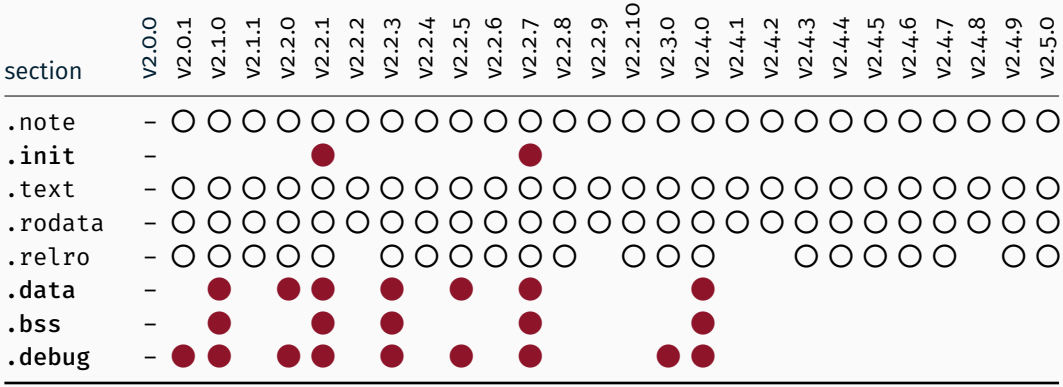
- from official source
- with default configuration and tools (`./buildconf.sh ; make`)
- no artifacts from previous builds
- using minimal Debian Bullseye environment

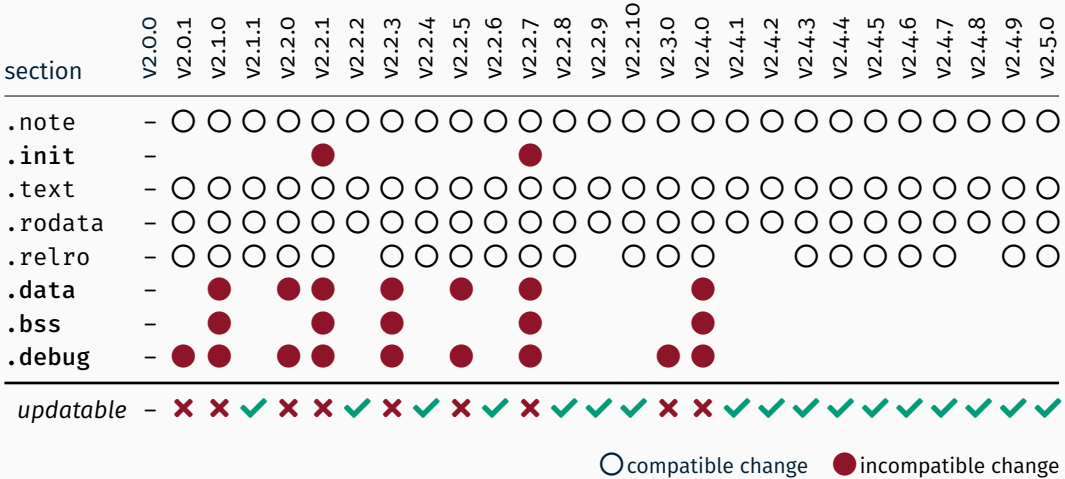

v2.0.0
v2.0.1
v2.1.0
v2.1.1
v2.2.0
v2.2.1
v2.2.2
v2.2.3
v2.2.4
v2.2.5
v2.2.6
v2.2.7
v2.2.8
v2.2.9
v2.2.10
v2.3.0
v2.4.0
v2.4.1
v2.4.2
v2.4.3
v2.4.4
v2.4.5
v2.4.6
v2.4.7
v2.4.8
v2.4.9
v2.5.0

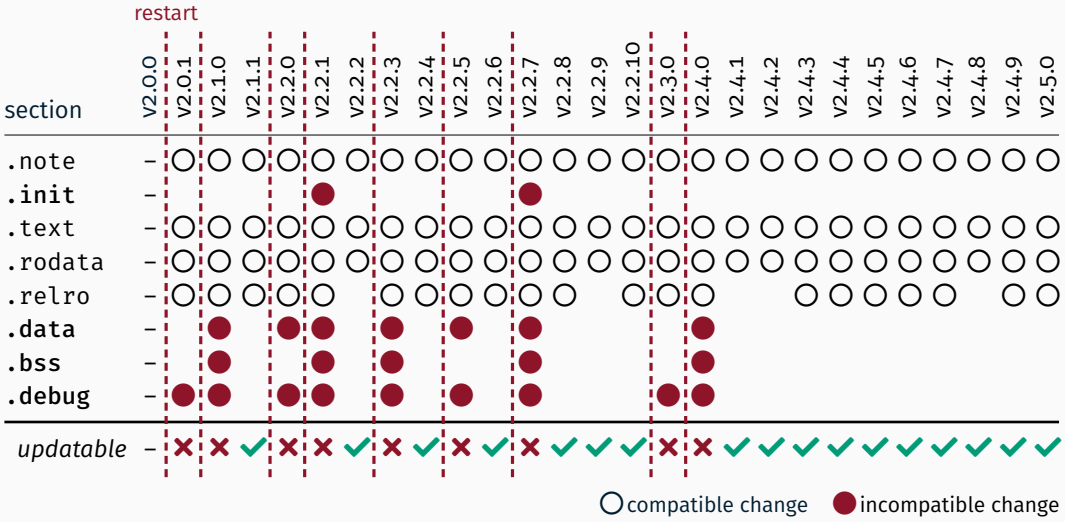
section	v2.0.0	v2.0.1	v2.1.0	v2.1.1	v2.2.0	v2.2.1	v2.2.2	v2.2.3	v2.2.4	v2.2.5	v2.2.6	v2.2.7	v2.2.8	v2.2.9	v2.2.10	v2.3.0	v2.4.0	v2.4.1	v2.4.2	v2.4.3	v2.4.4	v2.4.5	v2.4.6	v2.4.7	v2.4.8	v2.4.9	v2.5.0
.note	–	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

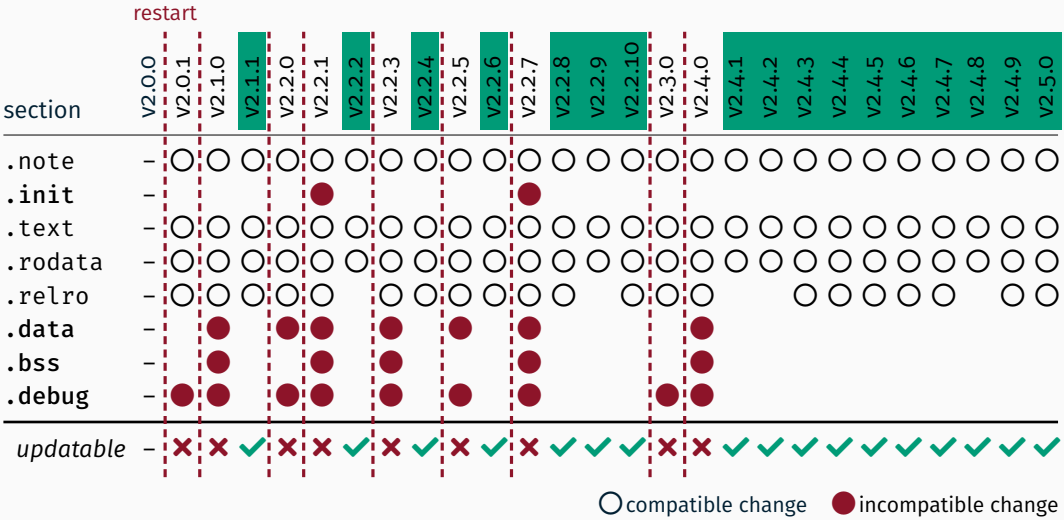
○ compatible change


section
v2.0.0
v2.0.1
v2.1.0
v2.1.1
v2.2.0
v2.2.1
v2.2.2
v2.2.3
v2.2.4
v2.2.5
v2.2.6
v2.2.7
v2.2.8
v2.2.9
v2.2.10
v2.3.0
v2.4.0
v2.4.1
v2.4.2
v2.4.3
v2.4.4
v2.4.5
v2.4.6
v2.4.7
v2.4.8
v2.4.9
v2.5.0
.note
.init
compatible change
incompatible change


section
v2.0.0
v2.0.1
v2.1.0
v2.1.1
v2.2.0
v2.2.1
v2.2.2
v2.2.3
v2.2.4
v2.2.5
v2.2.6
v2.2.7
v2.2.8
v2.2.9
v2.2.10
v2.3.0
v2.4.0
v2.4.1
v2.4.2
v2.4.3
v2.4.4
v2.4.5
v2.4.6
v2.4.7
v2.4.8
v2.4.9
v2.5.0
.note
.init
.text
.rodata
.relro
.data
.bss
.debug
compatible change
incompatible change


section
v2.0.0
v2.0.1
v2.1.0
v2.1.1
v2.2.0
v2.2.1
v2.2.2
v2.2.3
v2.2.4
v2.2.5
v2.2.6
v2.2.7
v2.2.8
v2.2.9
v2.2.10
v2.3.0
v2.4.0
v2.4.1
v2.4.2
v2.4.3
v2.4.4
v2.4.5
v2.4.6
v2.4.7
v2.4.8
v2.4.9
v2.5.0
.note
.init
.text
.rodata
.relro
.data
.bss
.debug
updatable
compatible change
incompatible change


restart
section
v2.0.0
v2.0.1
v2.1.0
v2.1.1
v2.2.0
v2.2.1
v2.2.2
v2.2.3
v2.2.4
v2.2.5
v2.2.6
v2.2.7
v2.2.8
v2.2.9
v2.2.10
v2.3.0
v2.4.0
v2.4.1
v2.4.2
v2.4.3
v2.4.4
v2.4.5
v2.4.6
v2.4.7
v2.4.8
v2.4.9
v2.5.0
.note
.init
.text
.rodata
.relro
.data
.bss
.debug
updatable
compatible change
incompatible change


restart
section
v2.0.0
v2.0.1
v2.1.0
v2.1.1
v2.2.0
v2.2.1
v2.2.2
v2.2.3
v2.2.4
v2.2.5
v2.2.6
v2.2.7
v2.2.8
v2.2.9
v2.2.10
v2.3.0
v2.4.0
v2.4.1
v2.4.2
v2.4.3
v2.4.4
v2.4.5
v2.4.6
v2.4.7
v2.4.8
v2.4.9
v2.5.0
.note
.init
.text
.rodata
.relro
.data
.bss
.debug
updatable
compatible change
incompatible change

Test Application

EXPAT test suite

- good coverage
- well-maintained

Test Application

EXPAT test suite

- good coverage
- well-maintained

} good evaluation target for dynamic updating releases

Test Application

EXPAT test suite

- good coverage
- well-maintained

} good evaluation target for dynamic updating releases

Test application

- using latest test suite from **EXPAT v2.5.0** release
- version-based skipping of incompatible / fatal tests (e.g., causing `segfault`)
- running in endless loop

EXPAT test suite

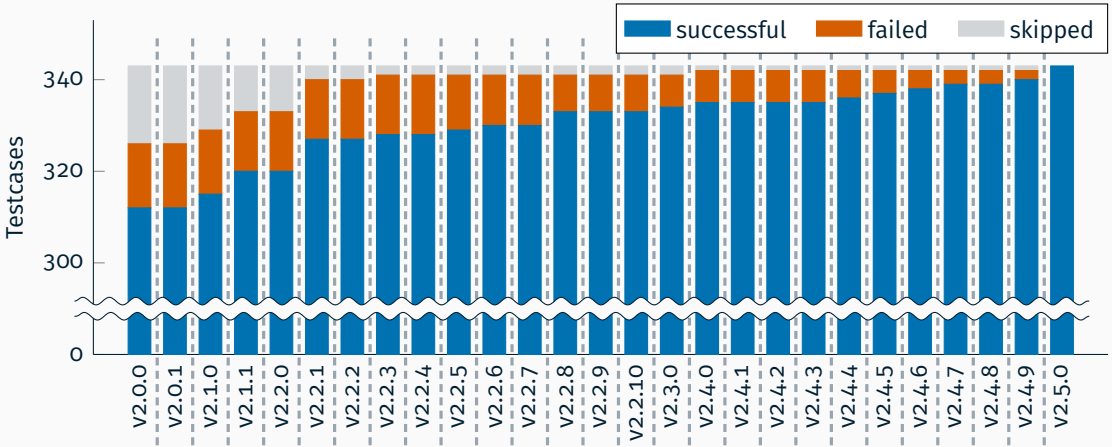
- good coverage
- well-maintained

} good evaluation target for dynamic updating releases

Test application

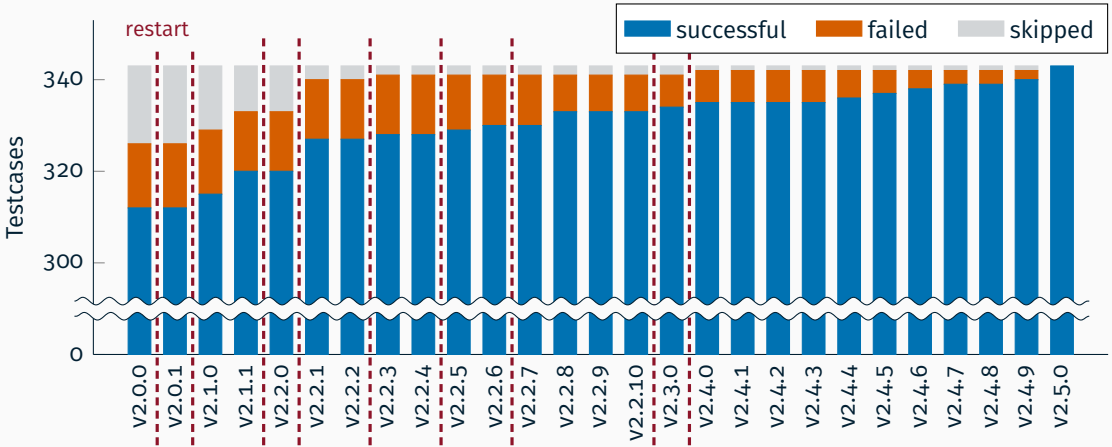
- using latest test suite from **EXPAT V2.5.0** release
- version-based skipping of incompatible / fatal tests (e.g., causing `segfault`)
- running in endless loop
- *no adjustments for dynamic updates or whatsoever!*

Test Application



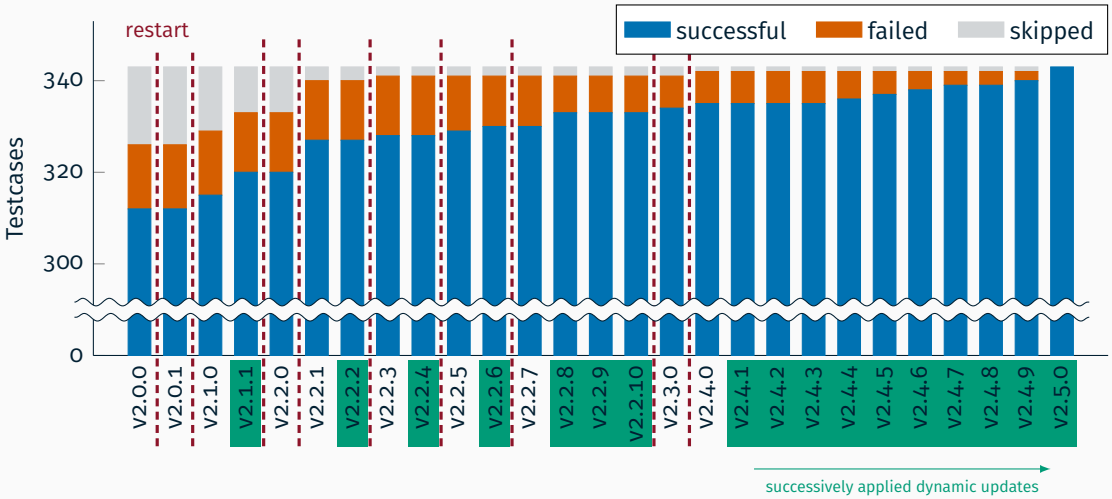
Test Application with Luci Dynamic Updates

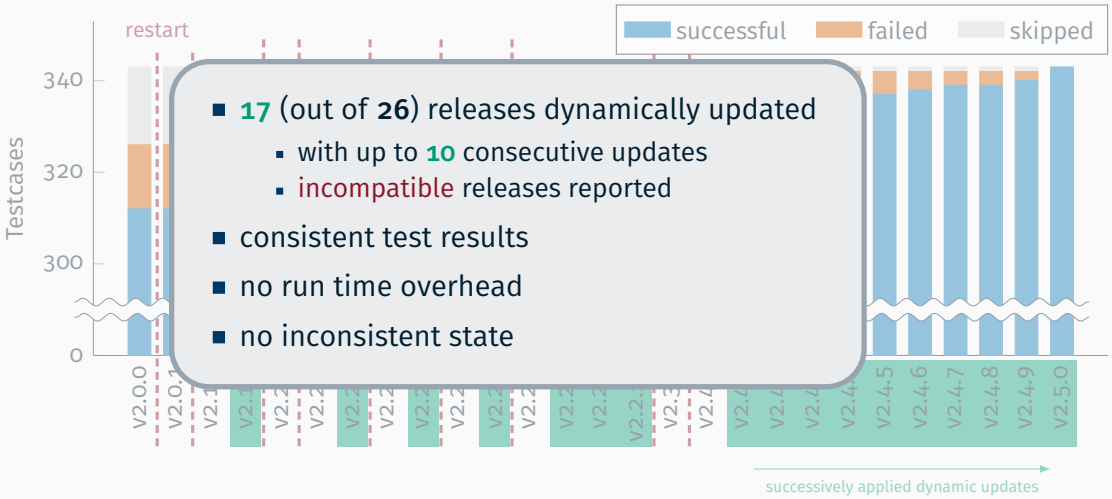
(vanilla) Expat



Test Application with Luci Dynamic Updates

(vanilla) Expat





- **17** (out of **26**) releases dynamically updated
  - with up to **10** consecutive updates
  - incompatible releases reported
- consistent test results
- no run time overhead
- no inconsistent state

Off-the-shelf Shared Objects

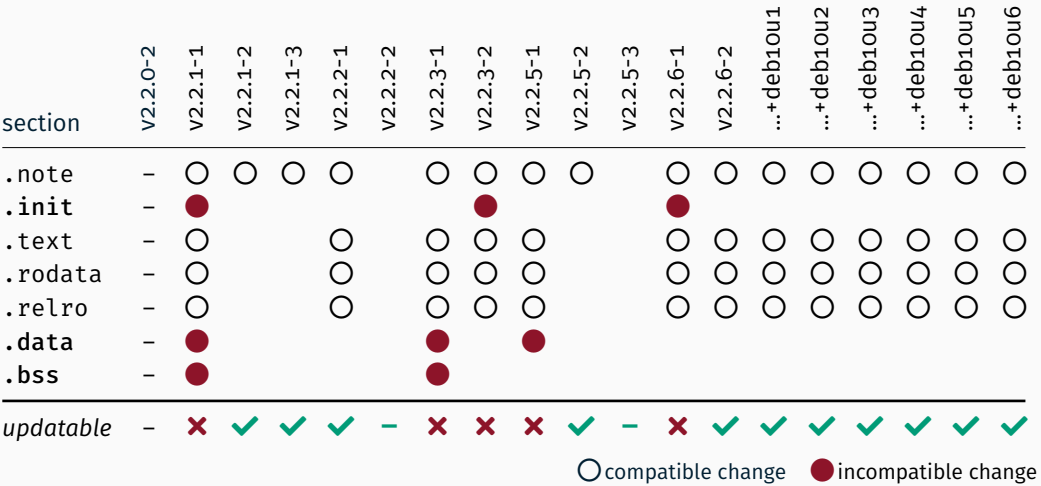
What about binary distributed libraries?

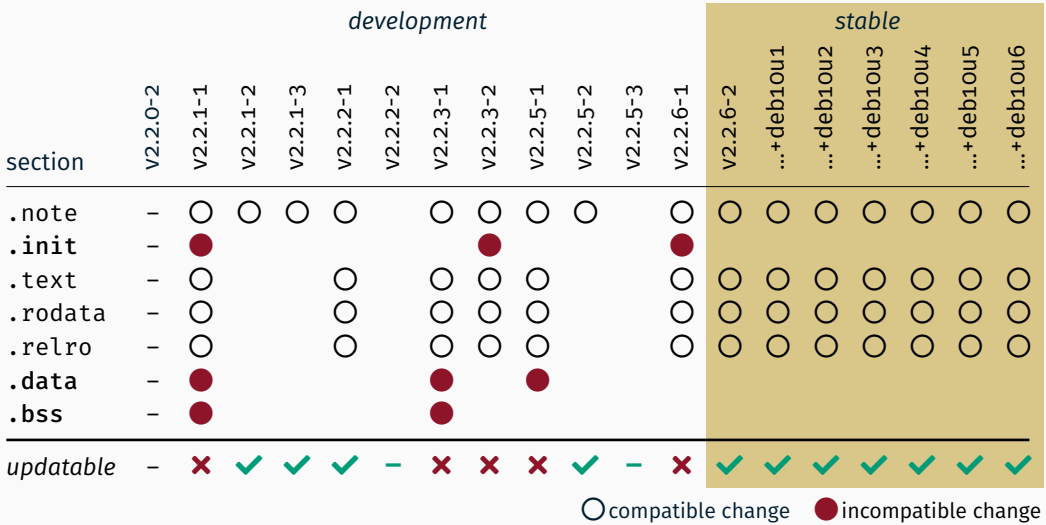
Off-the-shelf Shared Objects

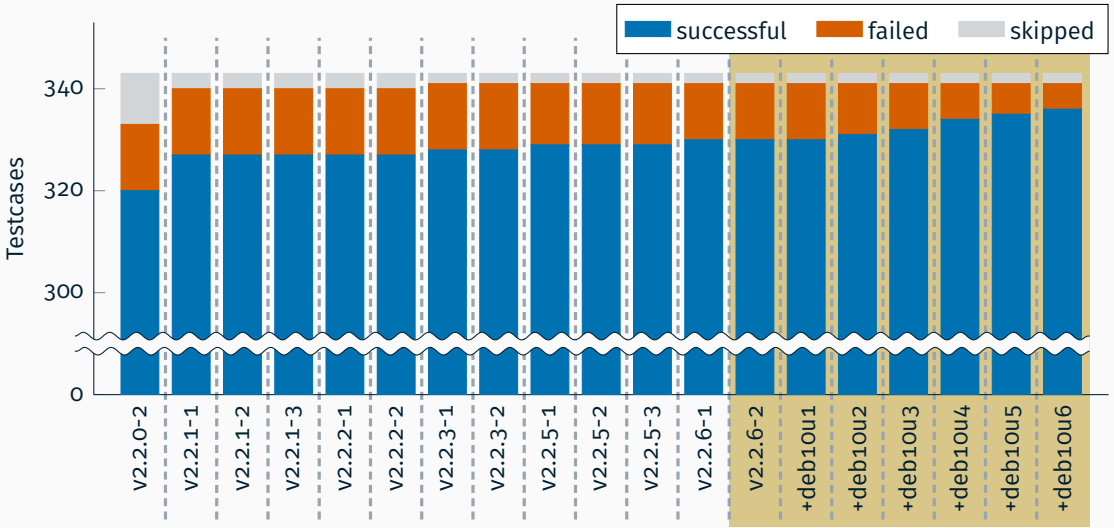
What about binary distributed libraries?

→ backtesting EXPAT shared objects from Debian Buster packages

- from Debian snapshot archive
- simulate previous real-world package updates
- including development builds
- excluding debug symbols

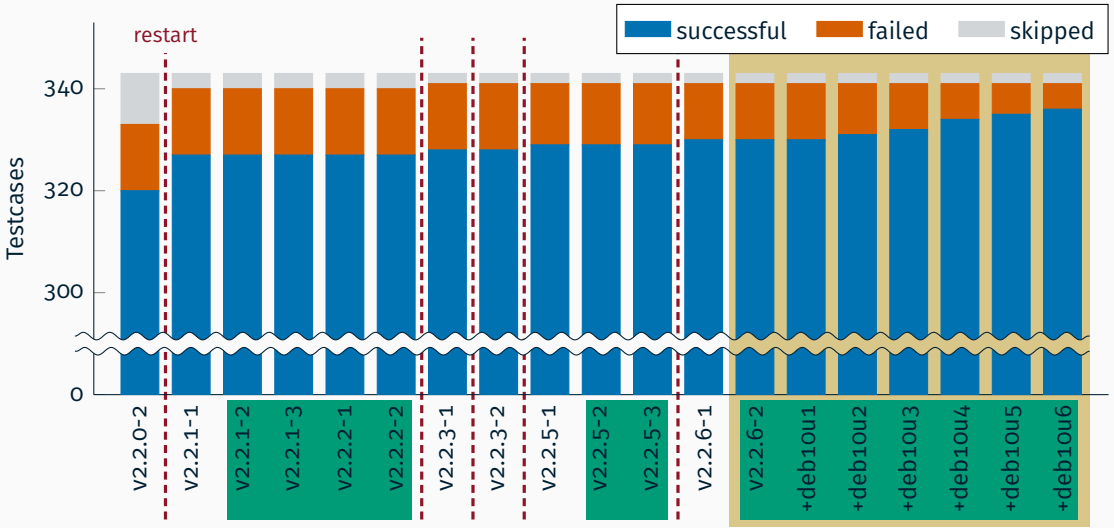

section
v2.2.0-2
v2.2.1-1
v2.2.1-2
v2.2.1-3
v2.2.2-1
v2.2.2-2
v2.2.3-1
v2.2.3-2
v2.2.5-1
v2.2.5-2
v2.2.5-3
v2.2.6-1
v2.2.6-2
...+deb10u1
...+deb10u2
...+deb10u3
...+deb10u4
...+deb10u5
...+deb10u6
.note
.init
.text
.rodata
.relro
.data
.bss
updatable
compatible change
incompatible change


development
stable
section
v2.2.0-2
v2.2.1-1
v2.2.1-2
v2.2.1-3
v2.2.2-1
v2.2.2-2
v2.2.3-1
v2.2.3-2
v2.2.5-1
v2.2.5-2
v2.2.5-3
v2.2.6-1
v2.2.6-2
...+deb10u1
...+deb10u2
...+deb10u3
...+deb10u4
...+deb10u5
...+deb10u6
.note
.init
.text
.rodata
.relro
.data
.bss
updatable
compatible change
incompatible change


successful
failed
skipped
Testcases
340
320
300
0
v2.2.0-2
v2.2.1-1
v2.2.1-2
v2.2.1-3
v2.2.2-1
v2.2.2-2
v2.2.3-1
v2.2.3-2
v2.2.5-1
v2.2.5-2
v2.2.5-3
v2.2.6-1
v2.2.6-2
+deb10u1
+deb10u2
+deb10u3
+deb10u4
+deb10u5
+deb10u6

Test Application with Luci Dynamic Updates

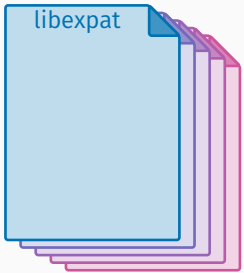
Expat from Debian Buster



Summary

Build		EXPAT	dynamic updates
custom (vanilla)		2.0.0 – 2.5.0	17 / 26 (65%)
Debian Buster	*all*	2.2.0 – 2.2.6	13 / 18 (72%)
	stable	2.2.6	6 / 6 (100%)
Debian Bullseye	*all*	2.2.7 – 2.2.10	9 / 10 (90%)
	stable	2.2.10	5 / 5 (100%)
Ubuntu Focal	*all*	2.2.7 – 2.2.9	6 / 6 (100%)
	stable	2.2.9	4 / 4 (100%)
Ubuntu Jammy	*all*	2.4.1 – 2.4.7	10 / 12 (83%)
	stable	2.4.7	2 / 2 (100%)

Evaluation Summary



Luci applying majority of releases without restart

- ✔ Expat XML parser
- ✔ libxcrypt (Extended Crypt Library)
- ✘ OpenSSL
- ✔ PECL (Perl 5 Compatible Regular Expression Library)
- ✔ Zlib

Conclusion

Dynamic linker/loader (RTLD)
with DSU-capabilities for libraries

- ☑ ready for off-the-shelf shared objects
  by adapting techniques for dynamic linking
- ☑ conforming to today's package distribution
- ☑ transparent to the user
- ☑ no runtime overhead
- ☑ sufficient for common bugfix changes

Source and Artifact available at
github.com/luci-project/eval-atc23

Thank you!