Elivepatch
Flexible distributed Linux Kernel live patching

Alice Ferrazzi

Summary

- Live patch explanation
- Current live patch services
  - Motivation for elivepatch
- Elivepatch solution
  - Implementation
  - Challenge
  - Status
  - Future Work
- Conclusion

kernel :~ $ whoami

- Gentoo
  - Gentoo Kernel Project Leader
  - Gentoo Kernel Security
  - Gentoo board member
  - Gentoo Google Summer of Code administrator and mentor for rust Gentoo project
- Cybertrust Japan
  - OSS Embedded Software Engineer
- Researcher
  - ACM SIGOPS member
  - Presented elivepatch as poster at SOSP 2017

This project was part of Google Summer of Code 2017 for the Gentoo organization.

Live patch explanation

Live patch

Modify the kernel without the need to reboot.

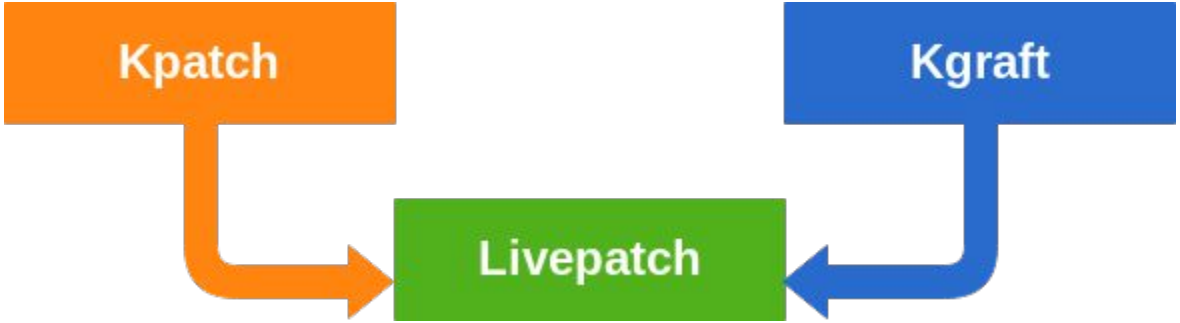
Why

- Downtime is expensive (containers, supercomputers)
- Security (vulnerability time shorter)

Where

- Embedded
- Mobile
- Desktops
- HPC (complex scientific computations)
- Cloud
- Any computer under heavy load

What



Kgraft

Suse Open Source live patching system that is routing the old function gradually.

Kpatch

Red Hat Open Source live patching system and use ftrace and stop_machine() for route functions toward the new function version.

Livepatch

Livepatch is a hybrid of kpatch and kgraft.

Livepatch has been merged into the kernel upstream.

Kpatch-build can work with both kpatch and livepatch for creating the live patch.

Livepatch is just a module

…

Livepatch module problem

A module that takes just about 1+ hour
to compile in a modern server

At Gentoo, we know what means to compile something for more than 1 hour…

Gentoo solution to compile for 1+ hour compilation problem

- Gentoo “binary host”
- Pre-compiled binary

What options do we have for compiling livepatch modules?

Current existing livepatch services

Current vendor solutions

- Oracle, Ksplice (support only Oracle Linux kernels)
- Suse Linux Enterprise Live Patching (support only Suse Kernels for one year)
- Canonical Live Patch (support only Ubuntu 16.04 LTS and Ubuntu 14.04 LTS)
- Red Hat live patch (Support only Red Hat kernel)

Motivation for elivepatch

Problems of vendor solutions

- **trusting on third-party** vendors
- Lacking support for **custom kernel configurations**
- Lacking support for **request-driven** costumization
- Lacking **long term** support
- **Closed source**

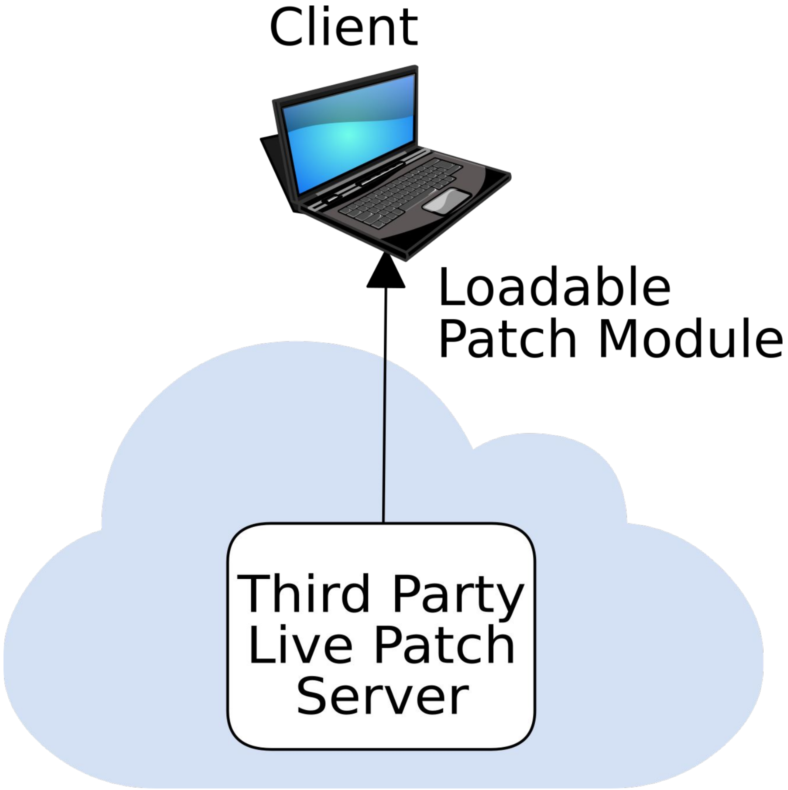
elivepatch solution

elivepatch

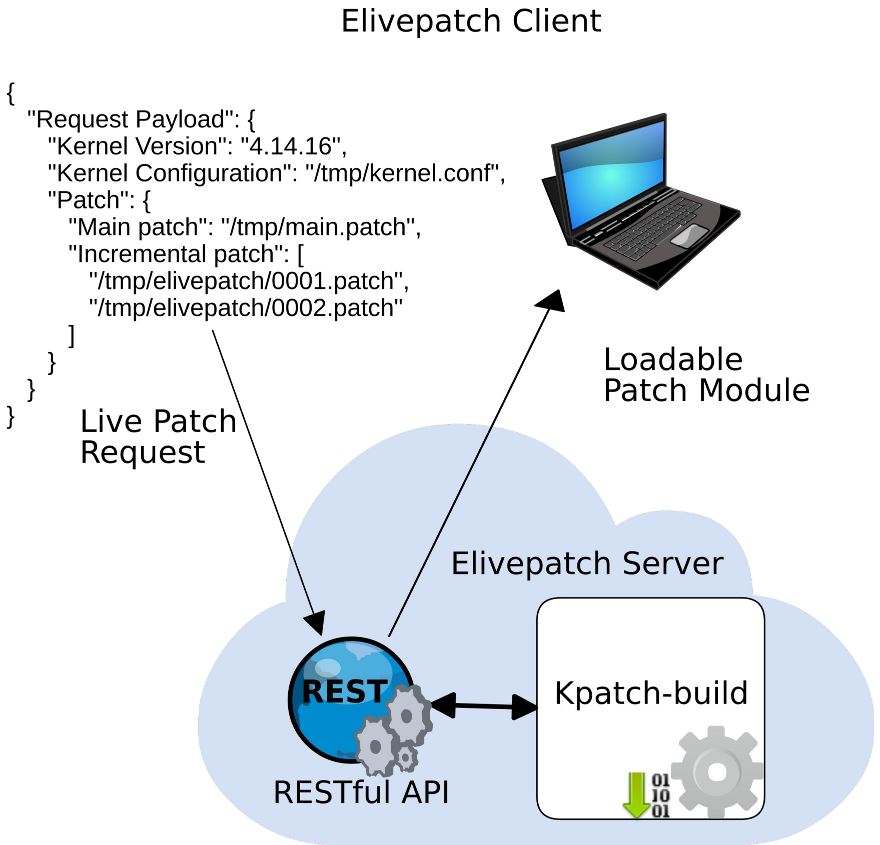
A web service framework to deliver Linux kernel live patches

- Supports **custom kernel configurations**
- User participation via **request-driven** customization
- **Open source**

Vendor solutions representation



Elivepatch solution



```
{
  "Request Payload": {
    "Kernel Version": "4.14.16",
    "Kernel Configuration": "/tmp/kernel.conf",
    "Patch": {
      "Main patch": "/tmp/main.patch",
      "Incremental patch": [
        "/tmp/elivepatch/0001.patch",
        "/tmp/elivepatch/0002.patch"
      ]
    }
  }
}
```

Implementation

Elivepatch-server (Main language: Python)

Flask + Flask-Restful + Werkzeug (not dependent)

Elivepatch-client (Main language: Python)

Requests + GitPython

Challenges

Challenges with elivepatch

- Some patches require manual modification to converted to live patches
- Reproducing the build environment can be difficult:
  - Differences in compiler versions
  - Variations in the compiler and optimization flags
- Incompatible machine architectures (solaris, hpc)

Incompatibility with GCC

CCFLAGS and non vanilla gcc,
can sometime broke elivepatch.

Current status

Elivepatch status

- First open source release 0.1 on 2017/9/06
- Packaged for Gentoo
- Presented as poster at SOSP 2017
- Close collaboration with kpatch mainteiners

Future work

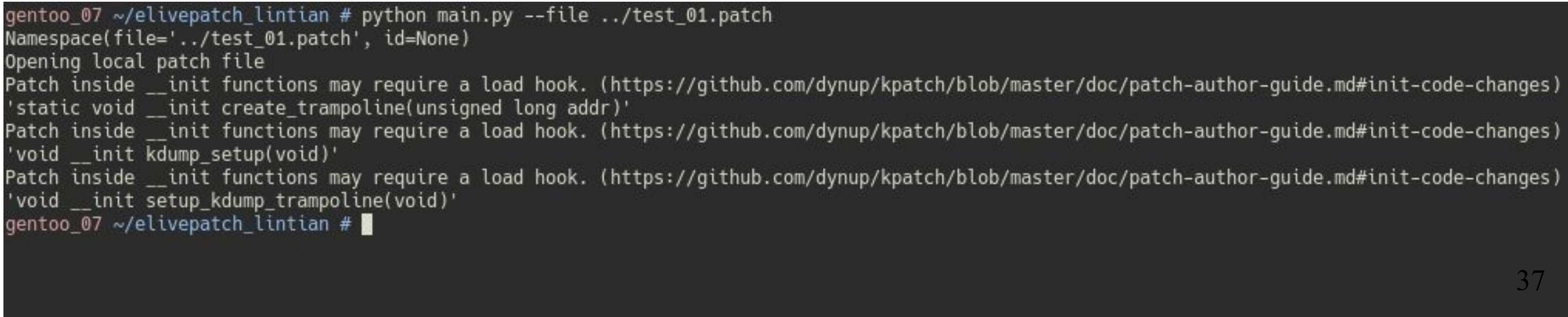
Future work

- Automate livepatch conversion
- Increasing scalability using containers and virtual machines
- Livepatch signing
- Kernel CI\CD check

Automate livepatch conversion

- Check patch for problems during conversion
- Suggest changes to patch for conversion
- Interest also for upstream to kpatch

https://github.com/aliceinwire/elivepatch_lintian



```
gentoo_07 ~/elivepatch_lintian # python main.py --file ../test_01.patch
Namespace(file='../test_01.patch', id=None)
Opening local patch file
Patch inside __init functions may require a load hook. (https://github.com/dynup/kpatch/blob/master/doc/patch-author-guide.md#init-code-changes)
'static void __init create_trampoline(unsigned long addr)'
Patch inside __init functions may require a load hook. (https://github.com/dynup/kpatch/blob/master/doc/patch-author-guide.md#init-code-changes)
'void __init kdump_setup(void)'
Patch inside __init functions may require a load hook. (https://github.com/dynup/kpatch/blob/master/doc/patch-author-guide.md#init-code-changes)
'void __init setup_kdump_trampoline(void)'
gentoo_07 ~/elivepatch_lintian #
```

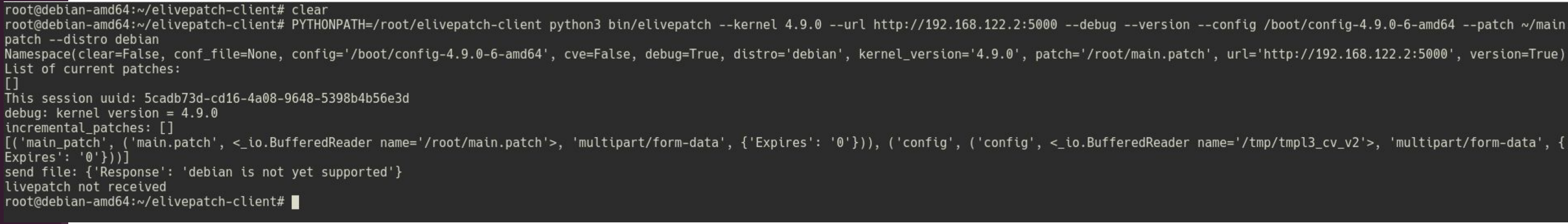
Multi distribution

Solve distributions compatibility issues

Current target:

- Debian
- Fedora
- Gentoo
- Android

Elivepatch client on Debian



```
root@debian-amd64:~/elivepatch-client# clear
root@debian-amd64:~/elivepatch-client# PYTHONPATH=/root/elivepatch-client python3 bin/elivepatch --kernel 4.9.0 --url http://192.168.122.2:5000 --debug --version --config /boot/config-4.9.0-6-amd64 --patch ~/main
patch --distro debian
Namespace(clear=False, conf_file=None, config='/boot/config-4.9.0-6-amd64', cve=False, debug=True, distro='debian', kernel_version='4.9.0', patch='/root/main.patch', url='http://192.168.122.2:5000', version=True)
List of current patches:
[]
This session uuid: 5cadb73d-cd16-4a08-9648-5398b4b56e3d
debug: kernel version = 4.9.0
incremental_patches: []
[('main_patch', ('main.patch', <_io.BufferedReader name='/root/main.patch'>, 'multipart/form-data', {'Expires': '0'})), ('config', ('config', <_io.BufferedReader name='/tmp/tmpl3_cv_v2'>, 'multipart/form-data', {
Expires': '0'}))]
send file: {'Response': 'debian is not yet supported'}
livepatch not received
root@debian-amd64:~/elivepatch-client# █
```

Work in progress…

https://asciinema.org/a/187738

p.s. Gentoo kernel is still needed

Livepatch signing

- Implementing livepatch module signing in the server
- Implementing signing verification for the client

Kernel CI/CD checking

- Implement a buildbot plugin for testing elivepatch

[You can test your livepatch with the same settings and hardware as where you want to deploy it]

Conclusion

Epilogue

- Live patch is a module that takes time compiling
- Live patch vendor service solutions solving the compilation problem
- Elivepatch solution

With the diffusion of embedded systems and robotics,

Livepatch services will become always more important

If you are interested in contributing,

Elivepatch is welcoming every form of contributions