Achieve a 0-downtime CERN Database infrastructure

August 2018

AUTHOR:
Varsha Rao

SUPERVISOR:
Borja Aparicio Cotarelo

CERN openlab

PROJECT SPECIFICATION

Key element at CERN that allows the laboratory to develop it is the mission and to deliver results is the ability to record, make available and keep safe the data collected for further analysis. Not only physics activities but a wide range of domains like accelerator controls, engineering or administrative systems require their data solutions provided by IT always available or with very few and short periods of maintenance with service interruption.

To offer reliability, availability and security at the same time is always a difficult task to achieve: there is no secure and reliable system if maintenance operations are not scheduled, which requires periods of downtime for interventions.

However, modern operating systems offer features that allow kernel patching without downtime like “Ksplice”, provided by Oracle and “Kpatch”, provided by RedHat.

Those tools would better allow the Database team (DB), apart from providing a higher level of availability without sacrificing reliability and security, to consolidate the infrastructure as data services would not require the level of partitioning implemented today.

The specific goals of these project, taking both products are:

- Test, evaluate and compare the technical features provided

- Evaluate and compare the effort needed to deploy those technologies in the DB systems

- Evaluate the operational impact of a platform where these tools are available compared with the current one

- Estimate the financial impact of acquiring any of those solutions compare with the current model (license cost)

Through this project, the candidate will acquire general knowledge about Linux based OS and Kernel and management techniques of a complex IT production environment including platform design and operations.

ABSTRACT

At CERN we have many systems which provide critical services and scheduling downtime for them is quite difficult. Live kernel patching is a technique which aims to update the system without any reboot or restart of the system. Using this approach, critical security updates can be applied to the systems at once without any disruption to the services. Oracle Ksplice and Red Hat Kpatch are two products which offer this solution. The aim of this project is to compare, evaluate and test these two products.

TABLE OF CONTENTS

Introduction 06

Dynamic Kernel Patching 07

Ksplice 08

- Installation Steps
- Test and Verify features

Kpatch 12

- Installation Steps
- Test and Verify features
- Limitations

Comparison of Ksplice and Kpatch 17

License Costs 18

Other Available Technologies (kGraft) 18

Conclusion 18

Reference 19

Acknowledgement

I would like to express my gratitude to my supervisor Borja Aparicio Cotarelo and IT-DB-IMS team for their constructive suggestions, guidance and support during my entire summer student internship period at CERN. I am also grateful to CERN Openlab for giving me the opportunity to work on this interesting project and in such a diverse work environment.

1. Introduction

With so many vulnerabilities being detected each day. It's important to keep the system up to date. Sometimes it's not possible to have downtime scheduled. The security vulnerabilities cost the companies a lot and even it causes inconvenience to the users. Using live kernel patching features, updates can be applied without having security compromised and downtime scheduled.

At CERN there are many critical systems for which scheduling downtime is difficult. With the rise in security vulnerabilities it is important to keep systems up to date. By the time downtime is scheduled, it happens that the system is still outdated due to the new vulnerabilities detected. It would be cheaper and efficient to have all the systems updated at once. Instead of separating them on the basis of service priorities and others.

Live Kernel patching is one of the ways in which systems can be kept updated without worrying much about downtime. It has the disadvantages and limitations. It is described in the upcoming sections.

The main objective of this Openlab project is to evaluate Ksplice with other available technologies in the market. We will understand, compare and evaluate Kpatch and Ksplice, offered by RedHat and Oracle respectively. Also determine the operational cost and feasibility of these products.

2. Dynamic Kernel Patching

Dynamic kernel patching/live kernel patching allows patches to be applied to a running kernel without restarting any process or rebooting the system. This technique is not something new, it has been around for a long time and still not widely used [1].

There are many products which provide this infrastructure using different approach such as Ksplice, Kpatch, Kgraft, Kexec and KernelCare. With many products available providing the solution, it is difficult to choose among them. There are some limitations associated with this technique, one of them is that it is not the same as kernel upgrade. The system needs to be rebooted for complete kernel upgrade.

Ksplice has been available for a long time. Initially it was an open source project, later acquired by Oracle. Both Red Hat and SUSE released their Open sourced version: Kpatch and kGraft almost around the same time [1]. The objective is the same for all of them but there are differences in the way they implement it. Red Hat and SUSE collaborated together for merging the live kernel patching features into the Linux Kernel Mainline.

3. Ksplice

It is a dynamic kernel patching program. This software was initially developed by four MIT students and later acquired by Oracle. Various security fixes are merged into the Linux Kernel each month. Ksplice updates are created using the kernel updates from Oracle or Linux Kernel community (see Figure 1).

.

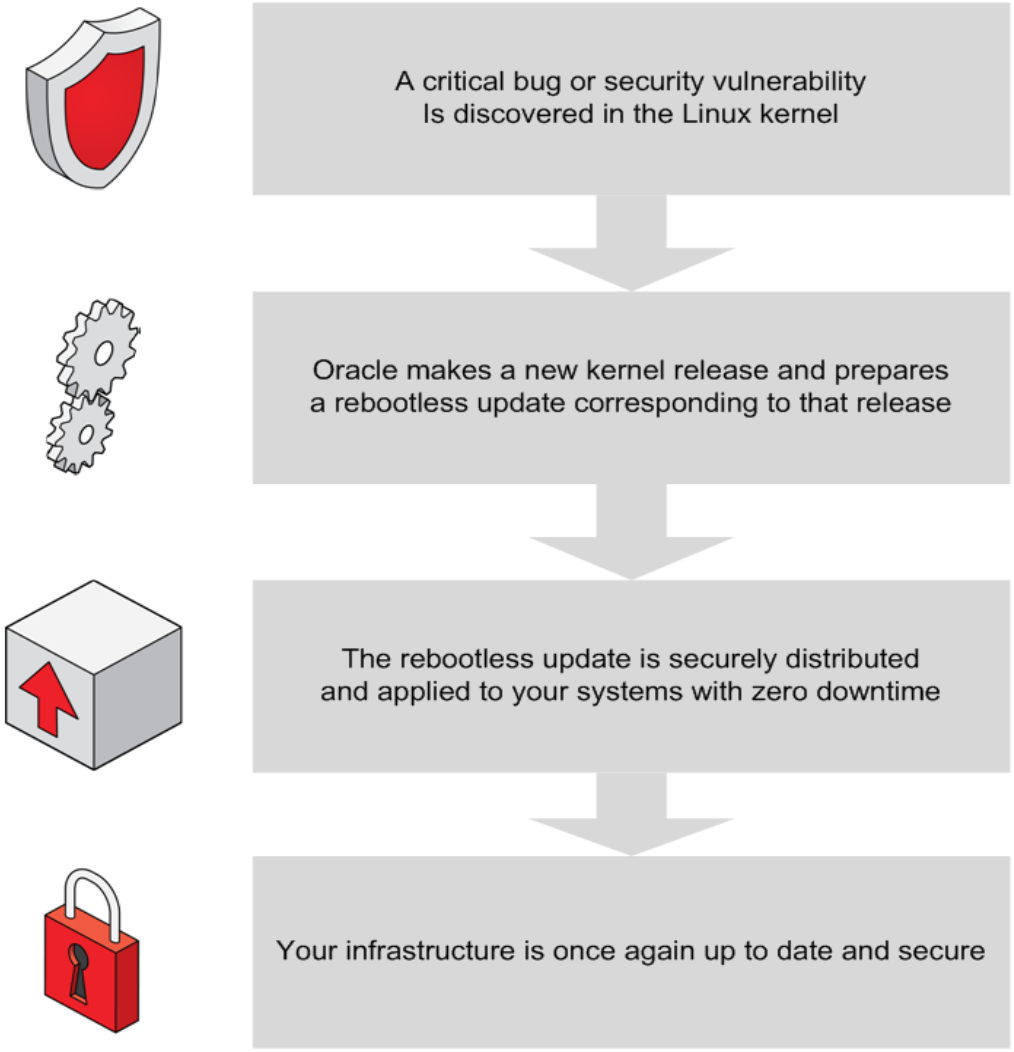

Figure 1. Life Cycle of Ksplice Update [2]

Working: It requires source tree of the Linux Kernel running on the system and security patch for it as the input. Two kernels are built, one with the patch and other one without it. It compares the resulting object files, to extract changed functions. After some processing, it is put into their own object file. Ksplice then creates a kernel module for this processed code and loads it into the kernel [3].

Use Cases: Runtime security vulnerabilities and stability bugs.
License: It was originally GPLV2, after Oracle acquisition it is no longer.
Merged into Kernel Mainline: No
Status: Production Ready

Installation: It's simple and quick.
Operating System: Support available only for Oracle Linux. Community Edition is provided for Fedora and Ubuntu.
Usage: The security patches are released by Oracle and can be applied easily. There is no transparency and flexibility with the patches. Only the patch's subject name can be viewed but not the actual changes. (see Figure 2)

```
[redacted]# uptrack-show --available
Available updates:
[fnk2i8lq] Workaround for alternative instruction inconsistencies.
[ij6564y4] Clear garbage data on the kernel stack when handling signals.
[hgdep9fv] Provide an interface to freeze tasks.
[k98laohw] Connection loss when forwarding between network namespaces.
```

Figure 2. Ksplice Usage Example

Some other features:

- It has more safety checks.
- It supports patching for *glibc*, *openssl* and other user space tools.
- It has web management console.

a. Installation Steps

Note: Run the commands as **root** user
There are two ways:

- Using the Installer:
  - Download *install-uptrack* script

```
wget -N https://www.ksplice.com/uptrack/install-uptrack
```

  - Run the *install-uptrack* script

    Note: Access key is required to run it. The key can be found under system status section in https://status-ksplice.oracle.com/status

```
[redacted]# sh install-uptrack <access-key>
```

- Manual Installation:
  - Download *Ksplice Uptrack* RPM package

```
[redacted]# wget https://ksplice.oracle.com/yum/uptrack/ol/ksplice-uptrack-release.noarch.rpm
```

  - Install the *uptrack* package

```
[redacted]# yum install ksplice-uptrack-release.noarch.rpm
# yum install uptrack
```

- Add access key

```
[redacted]# cat /etc/uptrack/uptrack.conf | grep accesskey
accesskey = <access-key>
```

This is the **common step** for both ways of installation.

- Update kernel

```
[redacted]# uptrack-upgrade -y
```

Installation steps are based on the Official Oracle Documentation [4].

b. Test and Verify Features

Below are the available Ksplice Commands [5].

- Check available patches to be applied

```
[redacted]# uptrack-show --available
Available updates:
[fnk2i8lq] Workaround for alternative instruction inconsistencies.
[ij6564y4] Clear garbage data on the kernel stack when handling signals.
[hgdep9fv] Provide an interface to freeze tasks.
[k98laohw] Connection loss when forwarding between network namespaces.

Effective kernel version is 3.10.0-862.6.3.el7
```

- Installing the updates

```
[redacted]# uptrack-upgrade -y
The following steps will be taken:
Install [fnk2i8lq] Workaround for alternative instruction inconsistencies.
Install [ij6564y4] Clear garbage data on the kernel stack when handling signals.
Install [hgdep9fv] Provide an interface to freeze tasks.
Install [k98laohw] Connection loss when forwarding between network namespaces.
Installing [fnk2i8lq] Workaround for alternative instruction inconsistencies.
Installing [ij6564y4] Clear garbage data on the kernel stack when handling signals.
Installing [hgdep9fv] Provide an interface to freeze tasks.
Installing [k98laohw] Connection loss when forwarding between network namespaces.
Your kernel is fully up to date.
Effective kernel version is 3.10.0-862.9.1.el7
```

Note:
The "uname -r" does not show the kernel version change from **3.10.0-862.6.3.el7** to **3.10.0-862.9.1.el7.** "uptrack-uname -r" displays the effective kernel version.

```
[redacted]# uname -r
3.10.0-862.6.3.el7.x86_64
[redacted]# uptrack-uname -r
3.10.0-862.9.1.el7.x86_64
```

- View the installed patches

```
[redacted]# uptrack-show
Installed updates:
[ij6564y4] Clear garbage data on the kernel stack when handling signals.
[hgdep9fv] Provide an interface to freeze tasks.
[fnk2i8lq] Workaround for alternative instruction inconsistencies.
[k98laohw] Connection loss when forwarding between network namespaces.

Effective kernel version is 3.10.0-862.9.1.el7
```

- We can use dmesg to verify the application of patches

```
[redacted]# dmesg | tail
[947886.652074] ksplice: Update fnk2i8lq applied successfully
[947886.857844] ksplice: Update ij6564y4 applied successfully
[947887.078608] ksplice: Update hgdep9fv applied successfully
[947887.556874] ksplice: Update k98laohw applied successfully
```

- Remove the installed updates

```
[redacted]# uptrack-remove --all -y
The following steps will be taken:
Remove [k98laohw] Connection loss when forwarding between network namespaces.
Remove [hgdep9fv] Provide an interface to freeze tasks.
Remove [ij6564y4] Clear garbage data on the kernel stack when handling signals.
Remove [fnk2i8lq] Workaround for alternative instruction inconsistencies.
Removing [k98laohw] Connection loss when forwarding between network namespaces.
Removing [hgdep9fv] Provide an interface to freeze tasks.
Removing [ij6564y4] Clear garbage data on the kernel stack when handling signals.
Removing [fnk2i8lq] Workaround for alternative instruction inconsistencies.
Effective kernel version is 3.10.0-862.6.3.el7
```

4. Kpatch

Kpatch developed by Red Hat, updates live kernel image without having to reboot the system. It makes use of the minimalistic Linux dynamic kernel patching infrastructure.

Working: The Kpatch way of working is similar to Ksplice. It performs the patching at functional level and checks the backtraces of all tasks to ensure that when a new function is applied, no instances of the old function are running. All functions are patched at the same time. This increases latency but it is safer with respect to data interactions [6]. Figure 3 provides an overview of Kpatch working.

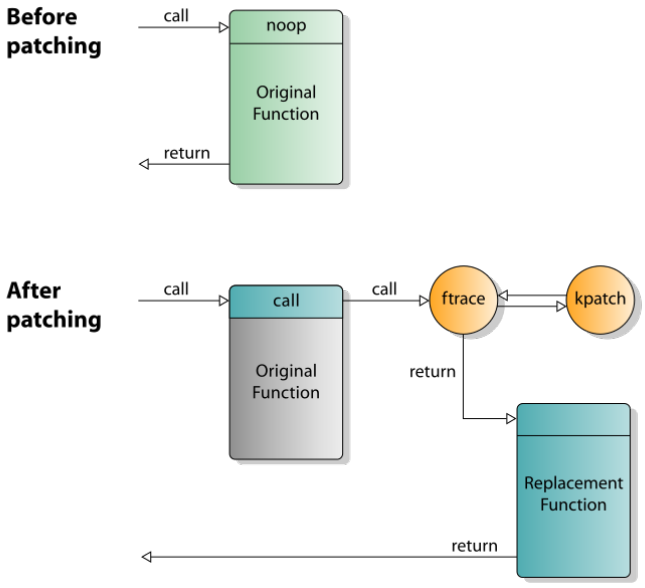

Figure 3. Live Kernel Patching by Kpatch Overview [7].

License: GPLV2
Merged into Kernel Mainline: Yes
Status: Not production ready
Installation: Easy and simple
Operating System: It is available on almost all major Linux distros i.e. Fedora, Ubuntu, CentOS, Debian and few others. Support is only available for RHEL.
Usage: Without support, the patch module needs to be built and loaded with two different commands. It provides transparency with the patches applied but takes time to create the patch, build and apply it. While creating a patch, Kpatch limitations needs to be taken into consideration. (See Figure 4 for example)

```
# Build Patch Module
[redacted]# kpatch-build -t vmlinux kpatch_version.patch
Using cache at /root/.kpatch/src
Testing patch file(s)
Reading special section data
Building original kernel
Building patched kernel
Extracting new and modified ELF sections
version.o: changed function: version_proc_show
Patched objects: vmlinux
Building patch module: livepatch-kpatch_version.ko
SUCCESS
# Load Module
[redacted]# kpatch load
loading patch module: livepatch-kpatch_version.ko
waiting (up to 15 seconds) for patch transition to complete...
transition complete (2 seconds)
```

Figure 4. Ksplice Usage Example

With support, the hotfixes patches are provided by Red Hat and can be applied easily.

```
# yum update kpatch-patch-7.0-2.el7.x86_64.rpm
```

a. Installation Steps

In this section, installation of Ksplice is described [8].

- Install dependencies for kpatch and kpatch-build.

```
[redacted]# yum install gcc kernel-devel-${UNAME%.*} elfutils elfutils-devel -y
[redacted]# yum install pesign yum-utils zlib-devel \
  binutils-devel newt-devel python-devel perl-ExtUtils-Embed \
  audit-libs audit-libs-devel numactl-devel pciutils-devel bison

[redacted]# yum-config-manager --enable debug
[redacted]# yum-builddep kernel-${UNAME%.*}

[redacted]# debuginfo-install kernel-${UNAME%.*}

# optional, but highly recommended - enable EPEL 7
[redacted]# yum install ccache
[redacted]# ccache --max-size=5G

# optional, for kpatch-test
[redacted]# yum install patchutils

#Additional packages required for compiling kernel
[redacted]# yum install java-devel ncurses-devel
```

- Clone the kpatch repository

```
[redacted]# git clone https://github.com/dynup/kpatch.git
```

- Build and install kpatch

```
[redacted]# make && make install
```

b. Test and Verify Features

Trying the available Kpatch commands.

- Copy kernel source code for creating the patch. There will be two copies of kernel source: the original (kernel_cp) and the modified (kernel_kpatch)

```
[redacted]# cp -r /usr/src/debug/kernel-4.9.el7.centos/linux-4.9.75-204.el7.centos.x86_64/* ~/kernel_cp
[redacted]# cp -r ~/kernel_cp/ ~/kernel_kpatch
```

- Create a test patch by modifying *fs/proc/version.c* file

```
[redacted]# vim kernel_kpatch/fs/proc/version.c
# Changes made in the patch
[redacted]# cat kpatch_version.patch
--- kernel_cp/fs/proc/version.c 2018-07-19 14:04:57.887692534 +0200
+++ kernel_kpatch/fs/proc/version.c     2018-07-19 15:29:12.588732987 +0200
@@ -8,6 +8,7 @@
 static int version_proc_show(struct seq_file *m, void *v)
 {
        seq_printf(m, linux_proc_banner,
+               "In version_proc_show",
                utsname()->sysname,
                utsname()->release,
                utsname()->version);
[redacted]# diff -u kernel_cp/fs/proc/version.c kernel_kpatch/fs/proc/version.c > kpatch_version.patch
```

- Create the patch module

```
[redacted]# kpatch-build -t vmlinux kpatch_version.patch
Using cache at /root/.kpatch/src
Testing patch file(s)
Reading special section data
Building original kernel
Building patched kernel
Extracting new and modified ELF sections
version.o: changed function: version_proc_show
Patched objects: vmlinux
Building patch module: livepatch-kpatch_version.ko
SUCCESS
```

- Next, load the built patch module

```
[redacted]# kpatch load
loading patch module: livepatch-kpatch_version.ko
waiting (up to 15 seconds) for patch transition to complete...
transition complete (2 seconds)
```

- Let's check the changes

```
[redacted]# cat /proc/version
In version_proc_show Linux (builder@kbuilder.dev.centos.org) (gcc version 4.8.5 20150623 (Red Hat 4.8.5-28) (GCC) ) 3.10.0-862.3.2.el7.x86_64
```

- Listing loaded modules

```
[redacted]# kpatch list
Loaded patch modules:
livepatch-kpatch_version [enabled]

Installed patch modules:
```

- Unloading the module

```
[redacted]# kpatch unload livepatch-kpatch_version.ko
disabling patch module: livepatch-kpatch_version
waiting (up to 15 seconds) for patch transition to complete...
transition complete (2 seconds)
unloading patch module: livepatch-kpatch_version
```

Some other commands to try out:

- Permanently installing the module

```
[redacted]# kpatch install livepatch-kpatch_version.ko
installing livepatch-kpatch_version.ko (3.10.0-862.3.2.el7.x86_64)
Created symlink from /etc/systemd/system/multi-user.target.wants/kpatch.service to /usr/local/lib/systemd/system/kpatch.service.
[redacted]# kpatch list
Loaded patch modules:

* Installed patch modules:
livepatch-kpatch_version (3.10.0-862.3.2.el7.x86_64)
```

- Checking patch information

```
[redacted]# kpatch info livepatch-kpatch_version.ko
Patch information for livepatch-kpatch_version.ko:
filename:       /root/livepatch-kpatch_version.ko
livepatch:      Y
license:        GPL
retpoline:      Y
rhelversion:    7.5
depends:
vermagic:       3.10.0-862.3.2.el7.x86_64 SMP mod_unload modversions
```

- Remove the installed module

```
[redacted]# kpatch uninstall livepatch-kpatch_version.ko
uninstalling livepatch-kpatch_version.ko (3.10.0-862.3.2.el7.x86_64)
```

Testing of commands were based on Kpatch documentation [8] and some other sources [9] [10].

c. Limitations [11] [12]

- It is not a general-purpose kernel upgrade mechanism.
- Not all CVE patches are supported such as one with data structure changes, modifications to init functions and some other cases.
- The system should not be suspended or hibernated when using kpatch.
- SystemTap or kprobe should not be used during or after loading a patch.

5. Comparison of Ksplice and Kpatch

In the Table 1, we compare the features provided by Ksplice and Kpatch.

	Ksplice	Kpatch
Developed By	Oracle	Red Hat
Initial Release	April 23, 2008	February 26, 2014
License	Initially GNU GPL V2	GNU GPL version 2
Use Case	Runtime security vulnerabilities and stability bugs.	Urgent security and stability fixes, CVEs, driver issues and kernel development.
Status	Production Ready	Not Production Ready
Installation	It can be done in 3 simple steps, using installer script.	Dependency needs to be taken care, the rest of the installation is easy.
Client Building Patches	Not possible	Possible
Operating System	Oracle Linux	Red Hat Enterprise Linux (RHEL)
Support	Available with premium users	Only premium users
Web Management Console	Available	Not Available
Merged in Kernel Mainline	No	Yes, since Kernel 4.1

Table 1. Comparing features of Ksplice and Kpatch

6. License Costs

Below are the licence type and cost estimates of Kpatch and Ksplice (Prices are in US Dollars for one year):

Kpatch

License Requirements: Red Hat Linux Premium SLA subscription and RHEL 7.2 onwards.
Cost: $1,299

Kslice

License Requirements: It is provided for free with a valid Oracle Linux Premier support contract.

Oracle Linux Premier Limited
License Price: -
Support Price: $1,399.00
Licensing Metric: System

Oracle Linux Premier
License Price: -
Support Price: $2,299.00
Licensing Metric: System

The terms and conditions with the license can be checked on the respective products official site [13] [14] [15].

7. Other available technologies (kGraft)

There are few other products available in the market, one of them is kGraft, which is developed by SUSE. It is also a live kernel patching feature available in the kernel. It does function based patching but it uses per task consistency. The old and new functions are allowed to run simultaneously and share data. Unlike Kpatch and Ksplice, it does not pause the system for applying patches. However, it's dangerous when the patch changes data structures. It requires SUSE Linux as the operating system [16] [17].

License: GNU GPL versions 2 and 3.

8. Conclusion

Ksplice and Kpatch cannot be used as an alternative for kernel upgrade. They are useful in case of applying some security patches. Ksplice requires a better reporting tool, the description of security fixes applied is not satisfactory. Due to criticality of the systems involved, both the products require a long-term evaluation.

9. References

1. Edge, J. "*An update on live kernel patching,"* Sept. 27, 2017. Available at: https://lwn.net/Articles/734765/ [Accessed Aug. 7, 2018]
2. Oracle, "*Oracle Linux Official Ksplice User Guide*". Available at: https://docs.oracle.com/cd/E37670_01/E39380/html/ol_about_ksplice.html [Accessed Aug. 14, 2018]
3. Arnold J. and Kaashoek, M. F. "*Ksplice: Automatic Rebootless Kernel Updates*". In Proceedings of the 4th ACM European conference on Computer systems (EuroSys'09), pages 187-198. April 2009 [Accessed July 10, 2018]
4. Oracle, "*Ksplice Official Installation Guide*". Available at: https://status-ksplice.oracle.com/status/installation [Accessed July 11, 2018]
5. Oracle, "*Uptrack User's Guide*". Available at: http://ksplice.oracle.com/uptrack/guide [Accessed July 11, 2018]
6. Corbet, J. "*The first kpatch submission,*" May 7, 2014. Available at: https://lwn.net/Articles/597407/ [Accessed 16 Jul. 2018]
7. W3C vector image," *With live patching in place, calls to patched kernel functions invoke their replacement counterparts*, "Nov. 11, 2014. https://commons.wikimedia.org/wiki/File:Linux_kernel_live_patching_kpatch.svg [Accessed Aug. 14, 2018]
8. Red Hat developers and other contributors. "*kpatch - live kernel patching,* "GitHub repository. Available at: https://github.com/dynup/kpatch [Accessed Jul 17, 2018]
9. Seidel, U. "*kpatch (1) - Linux Man Pages*". Available at: https://www.systutorials.com/docs/linux/man/1-kpatch/ [Accessed July 17, 2018]
10. Depuydt, J. "*Linux live kernel patching with kpatch on CentOS 7,"* July 28, 2015. Available at: http://jensd.be/651/linux/linux-live-kernel-patching-with-kpatch-on-centos-7 [Accessed July 17, 2018]
11. Red Hat, "*Kpatch official Documentation*". Available at: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/kernel_administration_guide/chap-documentation-kernel_administration_guide-working_with_kpatch [Accessed July 17, 2018]
12. Poimboeuf, J." *kpatch - Have your security and eat it too!*". Presented at LinuxCon North America, Aug. 22, 2014. Available at: https://events.static.linuxfound.org/sites/events/files/slides/kpatch-linuxcon_3.pdf [Accessed Aug. 14, 2018]
13. Oracle. "*Oracle Linux Support and Oracle VM Support Global Price List*". Available at: http://www.oracle.com/us/corporate/pricing/els-pricelist-070592.pdf [Accessed Sep. 6, 2018]
14. Oracle. "*Oracle Linux: Licensing Information User Manual for Release 6*". Available at: https://docs.oracle.com/cd/E37670_01/E63012/html/ol6-lic-restricted.html [Accessed Sep. 6, 2018]
15. Red Hat. "*Configure your Red Hat Enterprise Linux Server subscription"*. Available at: https://www.redhat.com/en/store/red-hat-enterprise-linux-server#?sku=RH00003 [Accessed Sep. 6, 2018]
16. SUSE, "*kGraft Official Site*". Available at: https://www.suse.com/products/live-patching/ [Accessed July 25, 2018]
17. Poimboeuf, J." kpatch*: dynamic kernel patching*," LWN.net, May 1, 2014**.** Available at: https://lwn.net/Articles/597123/ [Accessed July 25, 2018]