

Run John The Ripper on the cluster of Kali Linux leveraging MPI

利用 MPI 在 Kali Linux 集群上并行破解密码

By Ark

onlyarter@gmail.com

Github: @lfzark

2015.1

目录

1.配置环境与安装	3
1.1 所需环境和软件	3
1.2 安装过程	4
1.2.1 安装配置 MPICH	4
1.2.2 配置 SSH	7
1.2.3 安装支持 MPI 的 John The Ripper	9
2.性能比较测试	11
2.1 并行与非并行对比	11
2.2 进程数变化对比	12
3.利用 John+MPI 破解密码	13
3.1 利用 John+MPI 破解 raw-MD5	13
3.2 利用 John+MPI 破解 linux 密码	14

1.配置环境与安装

1.1 所需环境和软件

运行环境：

至少两台装有 kali-linux 的电脑。

kali-linux-1.0.7 内核版本:3.14。

两台电脑需要能互联,且需要有静态 IP , 可用 PING 命令测试。

所需软件与简介：

1. mpich2

MPI 是一个跨语言的通讯协议，用于编写并行计算机。支持点对点和广播。MPI 是一个信息传递应用程序接口，包括协议和语义说明，他们指明其如何在各种实现中发挥其特性。MPI 的目标是高性能，大规模性，和可移植性。MPI 在今天仍为高性能计算的主要模型。而 MPICH 是 MPI 标准的一种最重要的实现，可以免费从网上下载。MPICH 的开发与 MPI 规范的制订是同步进行的，因此 MPICH 最能反映 MPI 的变化和发展。

2. openssh

OpenSSH 是 SSH（Secure SHell）协议的免费开源实现。SSH 协议族可以用来进行远程控制，或在计算机之间传送文件。而实现此功能的传统方式，如 telnet(终端仿真协议)、rcp ftp、rlogin、rsh 都是极为不安全的，并且会使用明文传送密码。OpenSSH 提供了服务端后台程序和客户端工具，用来加密远程控件和文件传输过程中的数据，并由此来代替原来的类似服务。

3. john the ripper 1.7.1（或者 john the ripper 1.8.0）

John the Ripper 免费的开源软件，是一个快速的密码破解工具，用于在已知密文的情况下尝试破解出明文的破解密码软件，支持目前大多数的加密算法，如 DES、MD4、MD5 等。它支持多种不同类型的系统架构，包括 Unix、Linux、Windows、DOS 模式、BeOS 和 OpenVMS，主要目的是破解不够牢固的 Unix/Linux 系统密码。目前的最新版本是 John the Ripper 1.8.0 版，针对 Windows 平台的最新免费版为 John the Ripper 1.7.9 版。

1.2 安装过程

把两台主机分为一个主节点，一个从节点，配置基本相同,先从主节点为例开始配置

1.2.1 安装配置 MPICH

MPICH 跟大多数的 Linux 软件一样，有两种安装方式:apt-get 和源码编译安装

第一种比较简单，可以网上自行搜索，命令为

`sudo apt-get install mpich2` 不再赘述。

第二种是步骤如下

1. 下载源码包

```
root@localhost:~/Desktop# cp mpich2-1.0.2p1.tar.gz ~/
```

复制到当前用户的主目录（~指的是你当前用户的主目录，这里必须注意，两台主机需要配置相同的路径，否则会找不到文件）

2. 解压

```
root@localhost:~# tar -zxvf mpich2-1.0.2p1.tar.gz
```

3. 安装

```
root@localhost:~/mpich2-1.0.2p1# ./configure --prefix=/usr/mpich2-1.0.2
```

```
#/usr/mpich2-1.0.2 为安装目录
```

```
root@localhost:~/mpich2-1.0.2p1# make
```

```
root@localhost:~/mpich2-1.0.2p1# make install
```

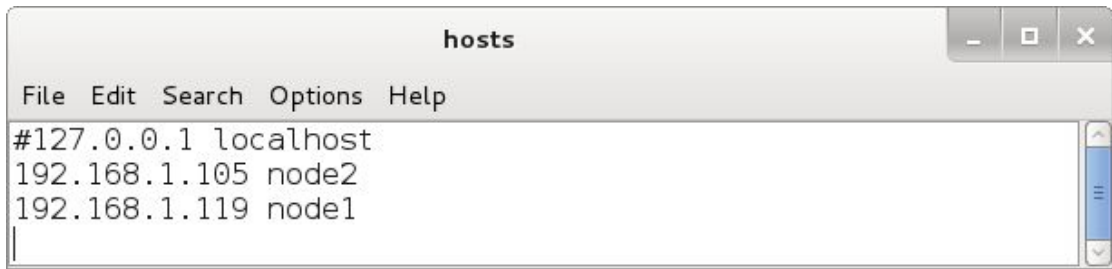
4. 配置 HOSTS 文件

```
root@localhost:~/mpich2-1.0.2p1# leafpad /etc/hosts
```

```
#127.0.0.1 localhost
```

```
192.168.1.119 node1 #为主节点在局域网 IP 地址
```

```
192.168.1.105 node2 #为从节点在局域网 IP 地址
```



```
hosts
File Edit Search Options Help
#127.0.0.1 localhost
192.168.1.105 node2
192.168.1.119 node1
```

配置 HOSTS 文件

5. 配置 MPICH

有两个配置文件需要建立

1. mpd.conf



```
mpd.conf
File Edit Search Options Help
secretword=cluster
```

root@localhost:~/mpich2-1.0.2p1# leafpad /etc/mpd.conf (kali 默认的是 leafpad 用不习惯安装个 gedit 或者使用 vi 都可以)

编辑内容为

Secretword=密码 --> 这个为集群密码所有节点的密码都需要一致才能进行集群通信

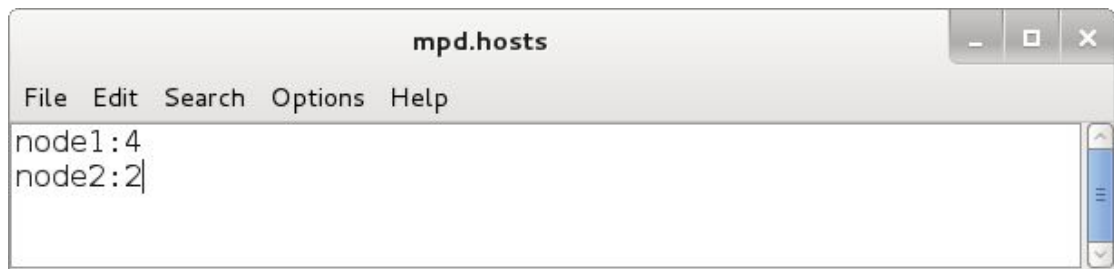
root@localhost:~# chmod 600 /etc/mpd.conf #这里必须设置 chmod 600

2. mpd.hosts

用到前面 HOSTS 文件里配置的信息

root@localhost:~/mpich2-1.0.2p1# leafpad /etc/mpd.hosts

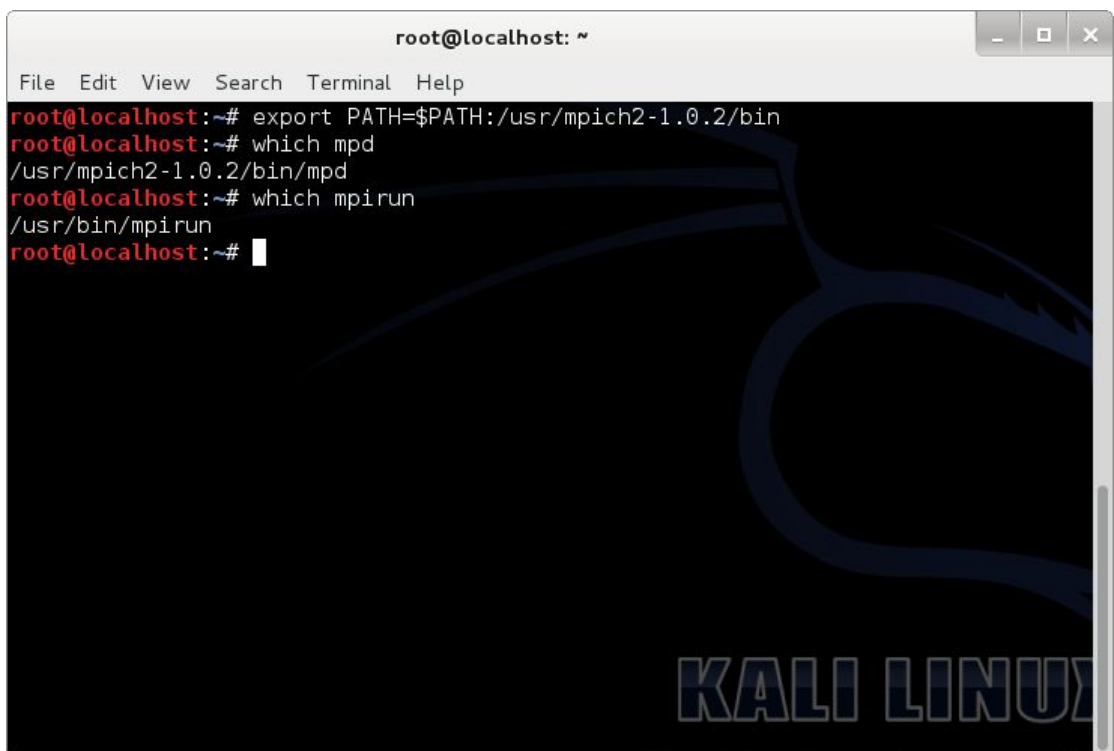
格式为 IP 地址: 所共享的 CPU 内核数



设置环境变量

```
export PATH=$PATH:/usr/mpich2-1.0.2/bin/
```

测试环境变量



1.2.2 配置 SSH

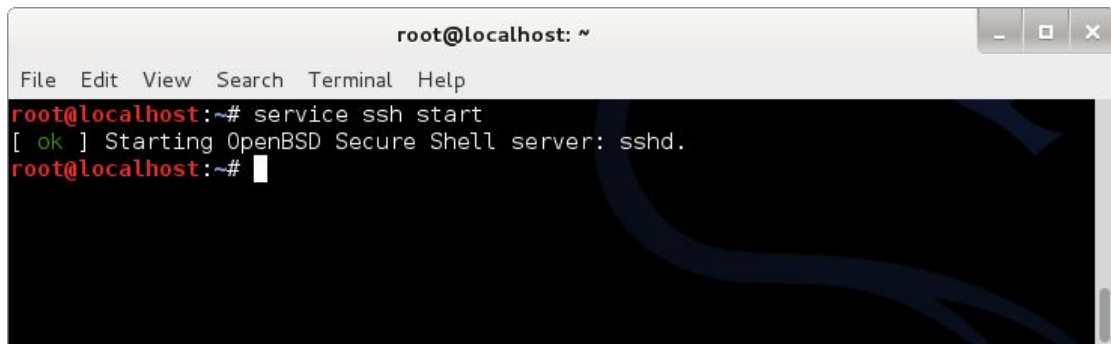
安装 openssh-server

```
root@localhost:~# apt-get install openssh-server
```

启动 ssh 服务

```
root@localhost:~# service ssh start
```

启动成功



SSH 免密码登录的方法

配置 SSH 免密码登录

```
root@localhost:~# ssh-keygen -t rsa    #生成.ssh 目录 主节点和从节点分别运行
```

```
root@localhost:~# scp node2:~.ssh/*  ~/.ssh    拷贝 node2 上的.ssh 目录到本地的.ssh
```

```
root@localhost:~# ssh node1
```

```
root@localhost:~# ssh node2
```

测试连接成功

到此已经配置完 MPICH 环境 测试一下

在主节点上设置

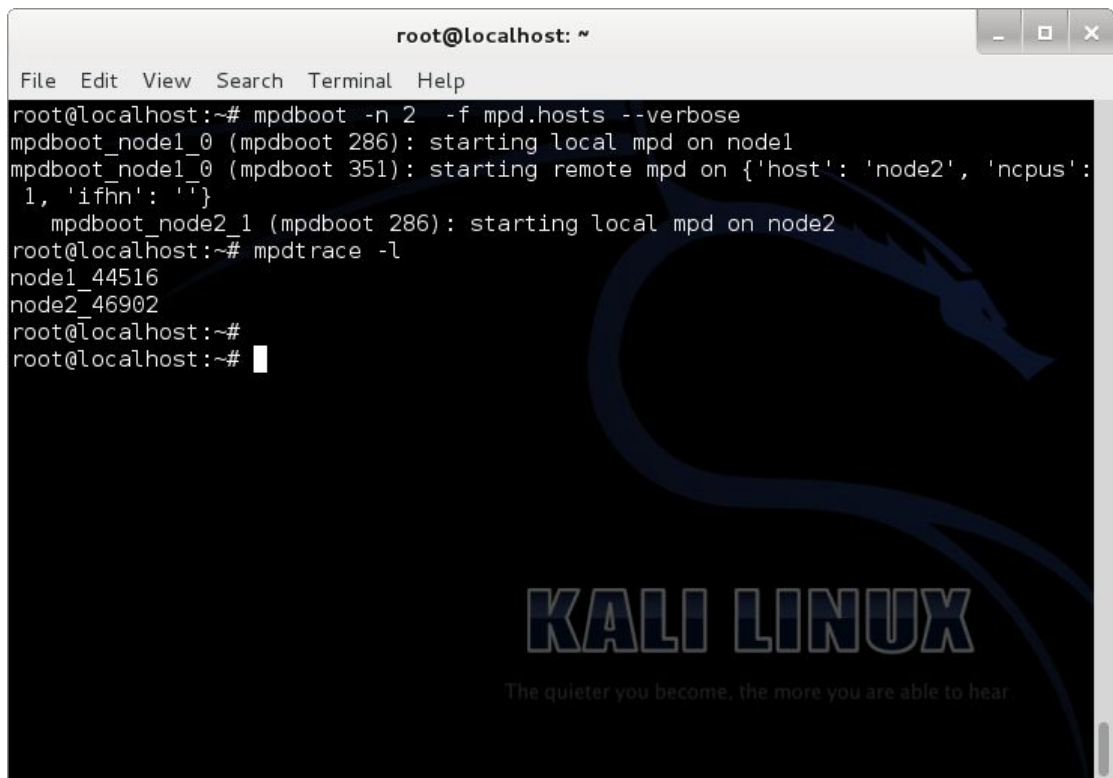
```
root@localhost:~# hostname node1
```

在从节点上设置

```
root@localhost:~# hostname node2
```

然后测试

```
root@localhost:~# mpdboot -n 2 -f mpd.hosts --verbose
```

A terminal window titled 'root@localhost: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal output shows the execution of 'mpdboot -n 2 -f mpd.hosts --verbose'. It reports starting local mpd on node1 (PID 286) and remote mpd on node2 (PID 351). Then, 'mpdboot_node2 1 (mpdboot 286): starting local mpd on node2' is shown. Finally, 'mpdtrace -l' is run, displaying 'node1_44516' and 'node2_46902'. The background of the terminal window features the Kali Linux logo and the text 'KALI LINUX' and 'The quieter you become, the more you are able to hear.'

```
root@localhost:~# mpdboot -n 2 -f mpd.hosts --verbose
mpdboot_node1_0 (mpdboot 286): starting local mpd on node1
mpdboot_node1_0 (mpdboot 351): starting remote mpd on {'host': 'node2', 'ncpus':
1, 'ifhn': ''}
    mpdboot_node2 1 (mpdboot 286): starting local mpd on node2
root@localhost:~# mpdtrace -l
node1_44516
node2_46902
root@localhost:~#
root@localhost:~#
```

```
root@localhost:~#Mpirun -n 2 hostname （输出主机名）
```

测试结果截图


```
root@localhost: ~/mpich2-1.0.2p1/bin
File Edit View Search Terminal Help
_IceTransSocketUNIXConnect: Cannot connect to non-local host localhost
(gedit:4107): EggSMClient-WARNING **: Failed to connect to the session manager:
Could not open network socket

root@localhost:~/mpich2-1.0.2p1/bin# ./mpdboot -n 2
mpdboot_node1_0 (mpdboot 393): error trying to start mpd(boot) at 1 {'host': 'node2', 'ncpus': 2, 'ifhn': ''}; output:
mpdboot_node2_1 (err_exit 415): mpd failed to start correctly on node2
reason: 1: invalid port from mpd configuration file /etc/mpd.conf is accessible by others
root@localhost:~/mpich2-1.0.2p1/bin# ./mpdboot -n 2
root@localhost:~/mpich2-1.0.2p1/bin# ./mpdtrace -l
node1_35188
node2_37244
root@localhost:~/mpich2-1.0.2p1/bin# ./mpirun -n 2 hostname
node1
node2
root@localhost:~/mpich2-1.0.2p1/bin# ./mpirun -n 4 hostname
node1
node2
node1
node2
root@localhost:~/mpich2-1.0.2p1/bin#
```

2.2.3 安装支持 MPI 的 John The Ripper

John 在 1.7.7-jumbo-5 版本开始已经支持 MPI，下面介绍一下支持 MPI 版本和 MPI 补丁版本各自的安装方式

MPI 补丁版本 例如 john-1.7.3.1-all-2-mpi8

编译 john the ripper

解压

```
root@node1:~/john-1.7.3.1-all-2-mpi8/src# make clean
```

```
root@node1:~/john-1.7.3.1-all-2-mpi8/src# make 选择合适的版本
```

这里 linux-x86-sse2 适合我的电脑 所以选择编译

```
root@node1:~/john-1.7.3.1-all-2-mpi8/src# make linux-x86-sse2
```

是否安装成功可进入 run 输入 ./john -test 进行测试

```
root@node1:~/john-1.7.3.1-all-2-mpi8# cd run/
```

```
root@node1:~/john-1.7.3.1-all-2-mpi8/john -test
```

```
root@node1: ~/john-1.7.3.1-all-2-mpi8/src
File Edit View Search Terminal Help
root@node1:~/john-1.7.3.1-all-2-mpi8/src# make clean
rm -f ../run/john ../run/unshadow ../run/unafs ../run/unique ../run/undrop ../ru
n/john.bin ../run/john.com ../run/unshadow.com ../run/unafs.com ../run/unique.co
m ../run/undrop.com ../run/john.exe ../run/unshadow.exe ../run/unafs.exe ../run/
unique.exe ../run/undrop.exe
rm -f ../run/john.exe john-macosx-* *.o *.bak core
rm -f detect bench generic.h arch.h sparc.h tmp.s
rm -f DES_bs_s.c DES_bs_n.c DES_bs_a.c
cp /dev/null Makefile.dep
root@node1:~/john-1.7.3.1-all-2-mpi8/src# make
To build John the Ripper, type:
make clean SYSTEM
where SYSTEM can be one of the following:
linux-x86-64      Linux, x86-64 with SSE2 (best)
linux-x86-sse2    Linux, x86 with SSE2 (best if 32-bit)
linux-x86-mmx     Linux, x86 with MMX
linux-x86-any     Linux, x86
linux-alpha      Linux, Alpha
linux-sparc      Linux, SPARC 32-bit
linux-ppc32-altivec Linux, PowerPC w/Altivec (best)
linux-ppc32      Linux, PowerPC 32-bit
linux-ppc64      Linux, PowerPC 64-bit
linux-ia64       Linux, IA-64
freebsd-x86-64   FreeBSD, x86-64 with SSE2 (best)
freebsd-x86-sse2 FreeBSD, x86 with SSE2 (best if 32-bit)
freebsd-x86-mmx  FreeBSD, x86 with MMX
freebsd-x86-any  FreeBSD, x86
freebsd-alpha    FreeBSD, Alpha
openbsd-x86-64   OpenBSD, x86-64 with SSE2 (best)
openbsd-x86-sse2 OpenBSD, x86 with SSE2 (best if 32-bit)
openbsd-x86-mmx  OpenBSD, x86 with MMX
openbsd-x86-any  OpenBSD, x86
openbsd-alpha    OpenBSD, Alpha
openbsd-sparc64  OpenBSD, SPARC 64-bit (best)
openbsd-sparc    OpenBSD, SPARC 32-bit
openbsd-ppc32    OpenBSD, PowerPC 32-bit
```

支持 MPI 版本，例如最新版本

在 Makefile 中修改如下

```
CC = mpicc -DHAVE_MPI -DJOHN_MPI_BARRIER -DJOHN_MPI_ABORT
```

```
MPIOBJ = john-mpi.o
```

也就是将原来的

改为

```
CC = mpicc -DHAVE_MPI -DJOHN_MPI_BARRIER -DJOHN_MPI_ABORT
```

并添加一行

```
MPIOBJ = john-mpi.o
```

```
root@node1:~/john-1.8.0/src# make clean
```

```
root@node1:~/john-1.8.0/src# make 选择合适的版本
```

是否安装成功可进入 run 输入./john -test 进行测试

最后测试一下 MPI+Joan 是否安装成功，成功的话，就可以进行接下来激动人心的并行密码破解了。

```
root@node1:~/john-1.8.0# cd run/
```

```
root@node1:~/john-1.8.0/john -test
```

john_mpi 测试

```
root@localhost: ~
File Edit View Search Terminal Help
root@localhost:~# cd john-1.7.3.1-all-2-mpi8/run/
root@localhost:~/john-1.7.3.1-all-2-mpi8/run# mpirun -n 2 ./john -test
Benchmarking: Traditional DES [128/128 BS SSE2]... DONE
Many salts: 4897K c/s real, 4897K c/s virtual

Only one salt: 4186K c/s real, 4186K c/s virtual

Benchmarking: BSDI DES (x725) [128/128 BS SSE2]... DONE
Many salts: 172288 c/s real, 172288 c/s virtual

Only one salt: 167936 c/s real, 167936 c/s virtual

Benchmarking: FreeBSD MD5 [32/32]... DONE
Raw: 14094 c/s real, 14094 c/s virtual

Benchmarking: OpenBSD Blowfish (x32) [32/32]... DONE
Raw: 561 c/s real, 563 c/s virtual

Benchmarking: Kerberos AFS DES [48/64 4K MMX]... ^CCtrl-C caught... cleaning up
processes
root@localhost:~/john-1.7.3.1-all-2-mpi8/run#
```

2.性能比较测试

2.1 并行与非并行对比

非 MPI	MPI
Benchmarking: Traditional DES [128/128 BS SSE2]... DONE	Benchmarking: Traditional DES [128/128 BS SSE2]... DONE
Many salts: 2700K c/s real, 2700K c/s virtual	Many salts: 8092K c/s real, 8176K c/s virtual
Only one salt: 2301K c/s real, 2301K c/s virtual	Only one salt: 6972K c/s real, 7044K c/s virtual
Benchmarking: BSDI DES (x725) [128/128 BS SSE2]... DONE	Benchmarking: BSDI DES (x725) [128/128 BS SSE2]... DONE
Many salts: 93952 c/s real, 93952 c/s virtual	Many salts: 280064 c/s real, 282282 c/s virtual
Only one salt:92800 c/s real, 92800 c/s virtual	Only one salt:271802 c/s real, 275322 c/s virtual
Benchmarking: FreeBSD MD5 [32/32]... DONE	Benchmarking: FreeBSD MD5 [32/32]... DONE
Raw: 7842 c/s real, 7842 c/s virtual	Raw: 25028 c/s real, 25303 c/s virtual
Benchmarking: Apache MD5 [32/32]... DONE	Benchmarking: Apache MD5 [32/32]... DONE
Raw: 7989 c/s real, 7989 c/s virtual	Raw: 25052 c/s real, 25272 c/s virtual
Benchmarking: Raw MD5 [raw-md5]... DONE	Benchmarking: Raw MD5 [raw-md5]... DONE
Raw: 6197K c/s real, 6260K c/s virtual	Raw: 19231K c/s real, 19397K c/s virtual

Benchmarking: Oracle [oracle]... DONE	Benchmarking: Oracle [oracle]... DONE
Raw: 800863 c/s real, 800863 c/s virtual	Raw: 2577K c/s real, 2612K c/s virtual
Benchmarking: MYSQL [mysql]... DONE	Benchmarking: MYSQL [mysql]... DONE
Raw: 3567K c/s real, 3567K c/s virtual	Raw: 10496K c/s real, 10600K c/s virtual

可以看出所有算法性能有了显著的提高。

2.2 进程数变化对比

N=5 MPI	N=4 MPI
Benchmarking: Traditional DES [128/128 BS SSE2]... DONE	Benchmarking: Traditional DES [128/128 BS SSE2]... DONE
Many salts: 8056K c/s real, 8210K c/s virtual	Many salts: 8092K c/s real, 8176K c/s virtual
Only one salt: 7106K c/s real, 7135K c/s virtual	Only one salt: 6972K c/s real, 7044K c/s virtual
Benchmarking: BSDI DES (x725) [128/128 BS SSE2]... DONE	Benchmarking: BSDI DES (x725) [128/128 BS SSE2]... DONE
Many salts: 280064 c/s real, 282277 c/s virtual	Many salts: 280064 c/s real, 282282 c/s virtual
Only one salt: 273536 c/s real, 276385 c/s virtual	Only one salt: 271802 c/s real, 275322 c/s virtual
Benchmarking: FreeBSD MD5 [32/32]... DONE	Benchmarking: FreeBSD MD5 [32/32]... DONE
Raw: 28899 c/s real, 29504 c/s virtual	Raw: 25028 c/s real, 25303 c/s virtual
Benchmarking: Apache MD5 [32/32]... DONE	Benchmarking: Apache MD5 [32/32]... DONE
Raw: 29117 c/s real, 29503 c/s virtual	Raw: 25052 c/s real, 25272 c/s virtual
Benchmarking: Raw MD5 [raw-md5]... DONE	Benchmarking: Raw MD5 [raw-md5]... DONE
Raw: 21429K c/s real, 22807K c/s virtual	Raw: 19231K c/s real, 19397K c/s virtual
Benchmarking: Oracle [oracle]... DONE	Benchmarking: Oracle [oracle]... DONE
Raw: 2919K c/s real, 2947K c/s virtual	Raw: 2577K c/s real, 2612K c/s virtual
Benchmarking: MYSQL [mysql]... DONE	Benchmarking: MYSQL [mysql]... DONE
Raw: 10107K c/s real, 10251K c/s virtual	Raw: 10496K c/s real, 10600K c/s virtual

上表可以看出有些算法随着进程数的增加并没有在性能上显著的提高。

3.利用 John+MPI 破解密码

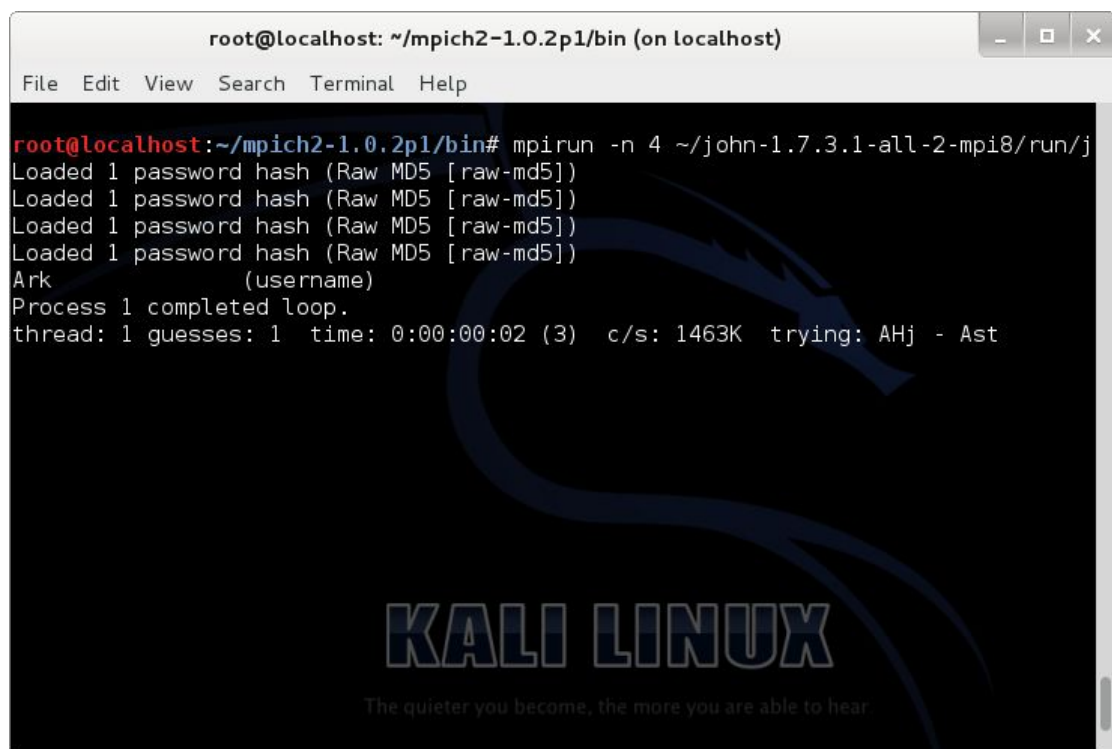
3.1 利用 John+MPI 破解 raw-MD5

```
root@node1:~/# echo username:efa4231e24c356d525a259f0b204404e > test.md5
```

从节点上需要配置相同的文件

```
root@node2:~/#scp -r node1:~/test.md5 ~/
```

```
root@node1:~/#mpirun -n 2 ~/john-1.7.3.1-all-2-mpi8/run/john --format=raw-MD5  
~/john-1.7.3.1-all-2-mpi8/run/test.md5
```




Raw-MD5 结果截图

3.2 利用 John+MPI 破解 linux 密码

```
root@node1:~/John/john-1.7.3.1-all-2-mpi8/run#tail -n 1 /etc/shadow >> linux.password  
root@node1:~/John/john-1.7.3.1-all-2-mpi8/run#cp linux.password ~/  
root@node1:~/mpich2-1.0.2p1/bin# ./mpirun -n 4 ~/john-1.8.0/run/john ~/linux.password
```

```
root@localhost: ~/mpich2-1.0.2p1/bin (on localhost)
File Edit View Search Terminal Help

root@localhost:~/mpich2-1.0.2p1/bin# ./mpirun -n 4 ~/john-1.8.0/run/john ~/linux.password
Loaded 1 password hash (crypt, generic crypt(3) [?/32])
No password hashes left to crack (see FAQ)
Crash recovery file is locked: /root/john-1.8.0/run/john.rec
Loaded 1 password hash (crypt, generic crypt(3) [?/32])
No password hashes left to crack (see FAQ)
Loaded 1 password hash (crypt, generic crypt(3) [?/32])
Press 'q' or Ctrl-C to abort, almost any other key for status
lg 0:00:00:01 100% 1/3 0.6134g/s 58.89p/s 58.89c/s 58.89C/s cluster..c999998
Use the "--show" option to display all of the cracked passwords reliably
Session completed
Loaded 1 password hash (crypt, generic crypt(3) [?/32])
cluster (cluster)
root@localhost:~/mpich2-1.0.2p1/bin#
root@localhost:~/mpich2-1.0.2p1/bin#
root@localhost:~/mpich2-1.0.2p1/bin#
```



KALI LINUX
The quieter you become, the more you are able to hear

破解 linux 密码结果截图

参考资料：

- 1.<http://openwall.info/wiki/john/parallelization>
- 2.<<John_the_Ripper_on_a_Ubuntu_10.04_MPI_Cluster>>

欢迎联系我共同讨论学习

Email : onlyarter@gmail.com