U.S. DEPARTMENT OF ENERGY | OFFICE OF Cybersecurity, Energy Security, and Emergency Response

Energy Storage Security (ESSec) Using Microservices Sandia National Laboratories

Adrian R. Chavez

Cybersecurity for Energy Delivery Systems (CEDS) Peer Review

October 6-7, 2020

Project Overview

Objective

- To design a secure and interoperable containerized suite of applications capable of testing, deploying, and operating an energy storage system. The team will include the ability to upgrade software in real-time, to quickly launch new applications, to detect compromised or crashed applications using fault tolerant algorithms, and to manage software applications, such as Open Field Message Bus protocols adapters, within a variety of energy storage systems.

Schedule

- Project start / end dates: (9/28/2020 - 9/29/2023)
- Research & Develop reference implementation (9/28/2020 - 3/31/2022)
- Partner sites (9/28/2020– 8/31/2023).
- Independent 3rd party red team assessment (3/31/2022-9/30/2022)
- Apply orchestration and container technologies towards utility

Total Value of Award:	**$3.75M**
Funds Expended to Date:	**0%**
Performer:	**Sandia National Laboratories**
Partners:	**Schweitzer Engineering Laboratories, Open Energy Solutions, Grimm, Duke Energy, DTE Energy, Entergy**

U.S. DEPARTMENT OF ENERGY | OFFICE OF Cybersecurity, Energy Security, and Emergency Response

Advancing the State of the Art (SOA)

- Energy Delivery Systems depend on software and management is manual:
  - Communicate/control resources
  - Optimize resources available
  - Maintenance and patching
  - Situational awareness – intelligent monitoring
  - BlackEnergy, Shamoon, and Stuxnet are examples of malware that target software applications to propagate through a control system network
- Currently, interruptions in service are necessary to update/upgrade software
- Application containers are used widely in IT environments but not in OT environments
- Virtual machines are heavyweight and not feasible for OT environments
  - Legacy systems are widely deployed in OT environments
- Software deployments are not portable

Advancing the State of the Art (SOA)

- Management of Energy Delivery Systems is a manual process
  - Configuration of software with existing infrastructure
  - Each deployment is unique
- Software containers reduce the attack surface by isolating processes within their own operating environments
- Orchestration allows for testing and repeatable deployments
- Software containers are portable across operating systems
- Processes compromised with malware can be identified and replaced without disruption of operation
- Interoperable solution that can be federated across multiple utilities
  - Commercial product will be included within SEL product line
- Testing, evaluation, and documentation of our approach

Progress to Date

Major Accomplishments

- 2020 - Kickoff meeting complete Monday, September 28, 2020
- 2020 – Developed live-upgrades and live-migration technology proof-of-concept (CAPSec)
- 2020 - DOE Practices to Accelerate the Commercialization of Technologies (PACT) (CAPSec)
- 2018 - US Patent No. 10,037,203 (CAPSec)

Challenges to Success

Containerization of energy storage system

- Identify software container candidates applicable for utility partner sites
- Identify ESS software applicable to each of the partner sites
- Build and deploy microservices within laboratory environment
- Measure performance of containerized energy storage system

Orchestration of containerization solution

- Identify software orchestration candidates to manage containers
- Develop microservices that can be upgraded and migrated without downtime
- Test and deploy within utility partner sites

Deployment within utility partner sites

- Each utility will have unique environments
- Orchestration and containerization will need to be portable and federated across partner sites
- Open Field Message Bus protocol and SEL commercial product will provide interoperability

Collaboration/Sector Adoption

Plans to transfer technology/knowledge to end user

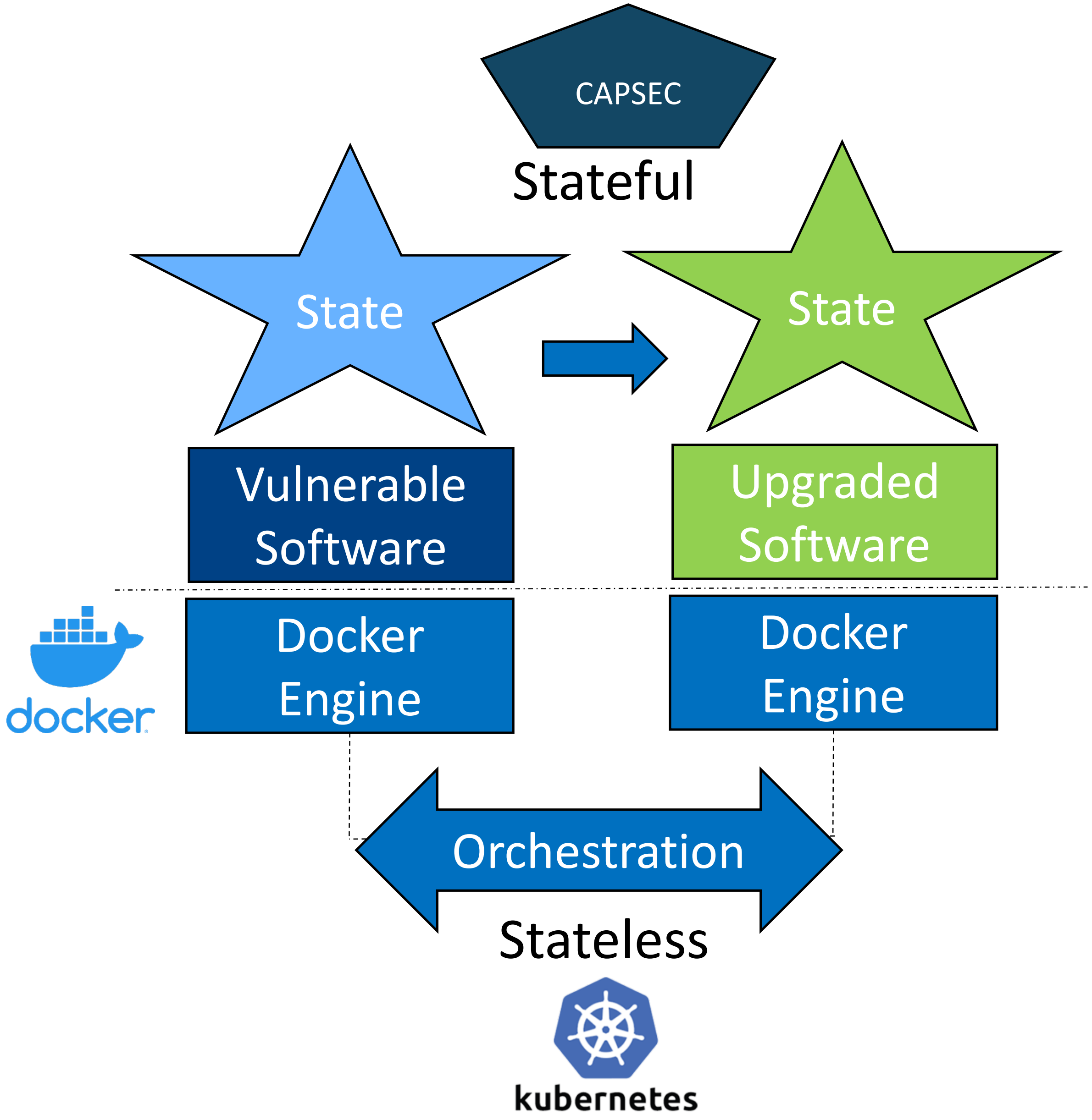
- Partnering with 3rd party red team throughout lifecycle of project
  - Provide cybersecurity guidance from design to deployment
- Partnering with vendor to commercialize solution
  - Reference implementation will be Interoperable commercial product
- Partnering with 3 utilities to test and evaluate our solutions
  - Ensure we are meeting functional and operational requirements
  - Develop use cases that are broadly applicable across each utility
  - Federate solution between each of the utility partners
  - Demonstration to be performed during 2nd half of the project after the reference implementation and the red team assessment are complete

Next Steps for this Project

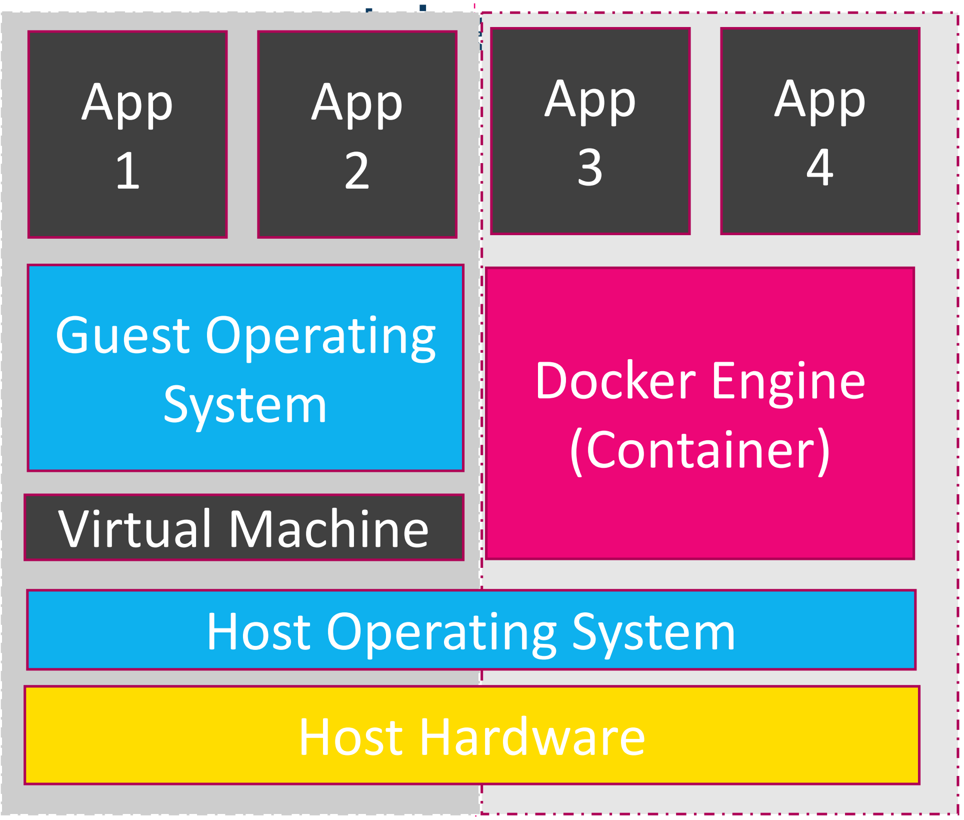
Approach for the next year or to the end of project

- Orchestration/Containerization of Energy Storage System (9/28/2020-3/31/2022)
  - Interoperable reference implementation to be developed
- Capture and document performance metrics (1/31/2022-7/3/2023)
  - Ensure operational requirements of utility partners are met
- Independent 3rd party red team security assessment (3/31/2022-9/30/2022)
  - Document findings and mitigations to be integrated into reference implementation
- Apply towards utility environments (9/30/2022-8/31/2023)
  - Integrate reference implementation and commercial product to demonstrate interoperability
- Document and communicate results to partners and DOE (9/28/2020-9/29/2023)

Container Application Security (CAPSec)

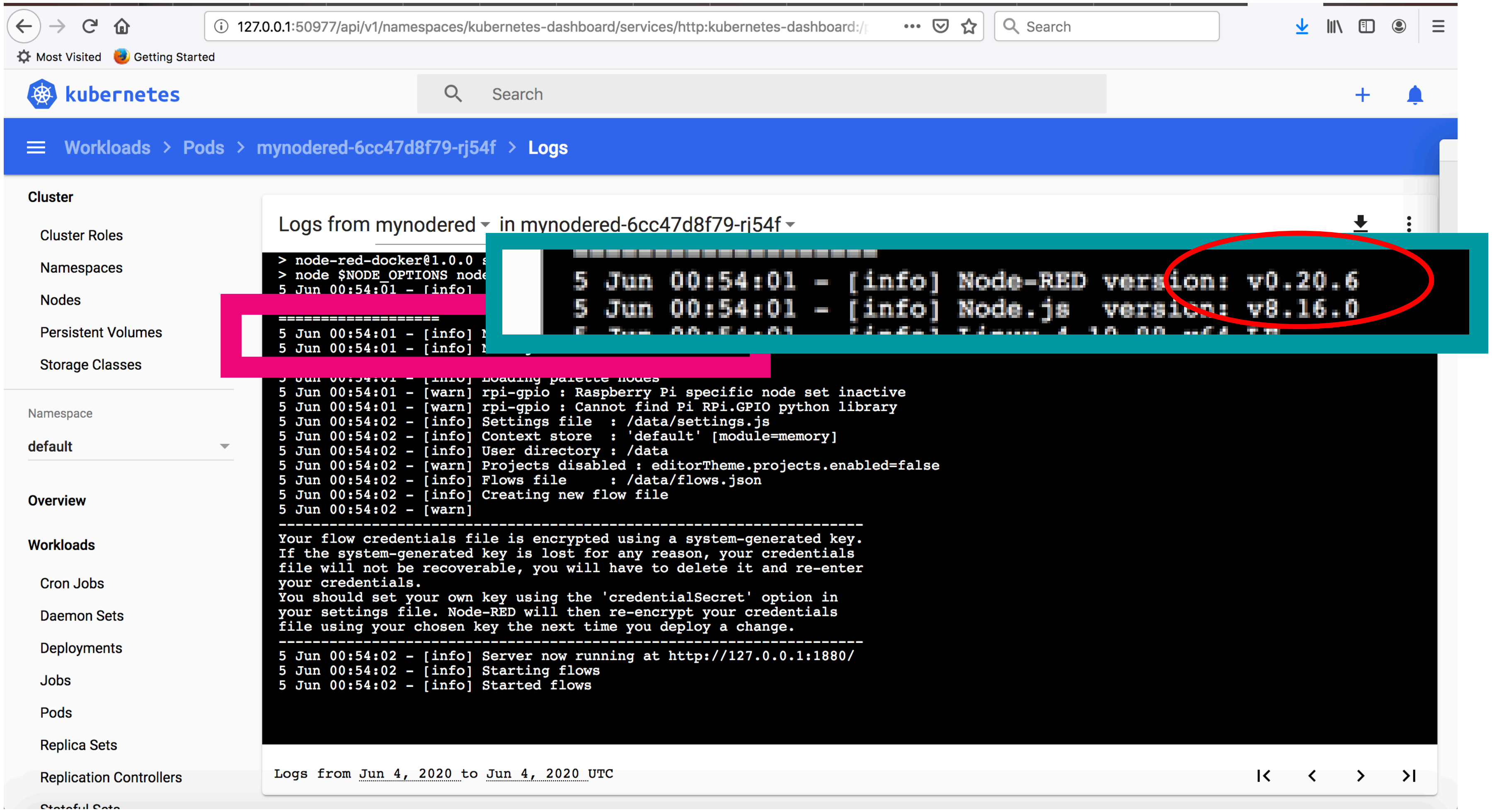


- Containerized architecture
- Replicates Stateful environments
- TLS Agents on parallel



Demonstration 1: NodeRed Software BEFORE Upgrade

- NodeRed = 0.20.6
- Node.js = 8.16.0



Logs from mynodered in mynodered-6cc47d8f79-rj54f

```
> node-red-docker@1.0.0 s
> node $NODE_OPTIONS node
5 Jun 00:54:01 - [info]

===================
5 Jun 00:54:01 - [info] N
5 Jun 00:54:01 - [info] N
```

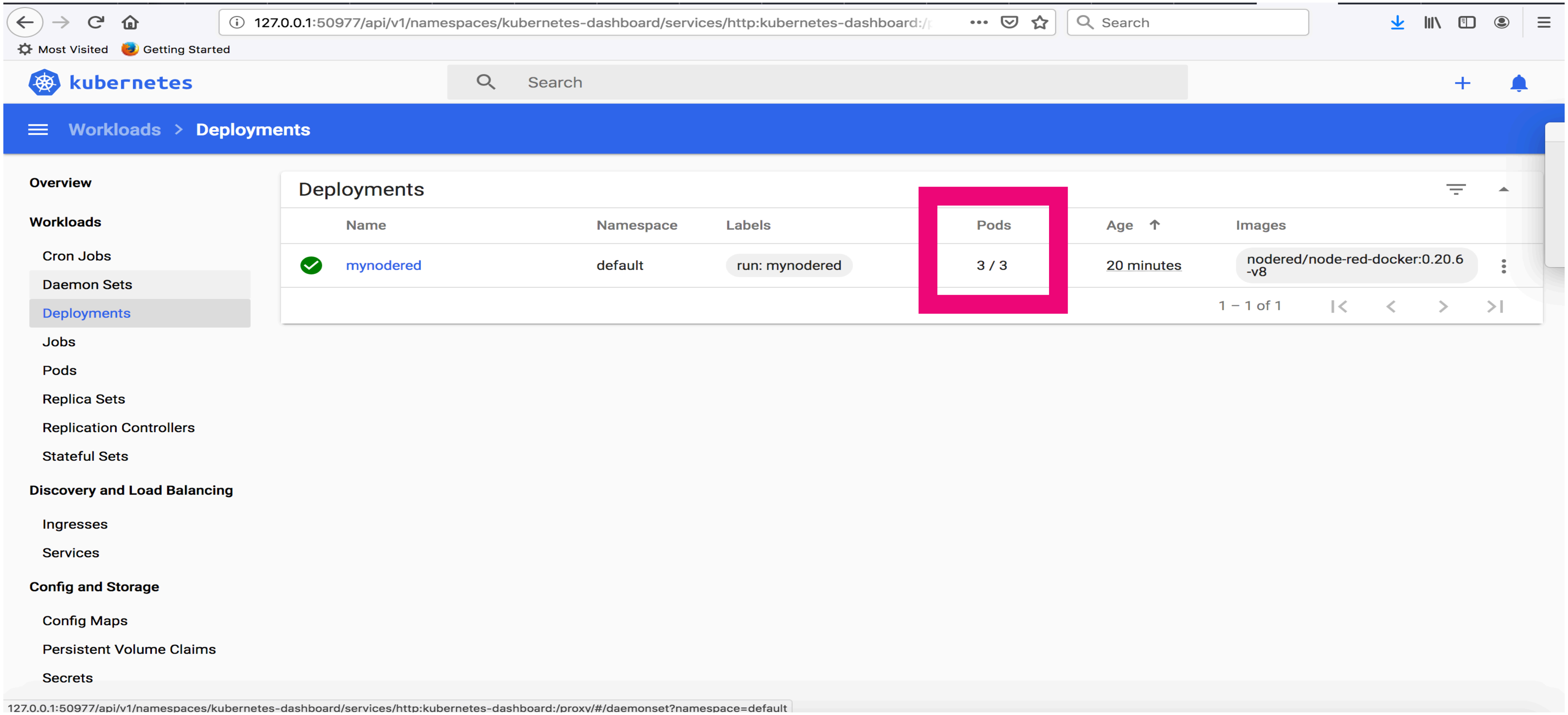
```
5 Jun 00:54:01 - [info] Node-RED version: v0.20.6
5 Jun 00:54:01 - [info] Node.js  version: v8.16.0
```

```
5 Jun 00:54:01 - [info] Loading palette nodes
5 Jun 00:54:01 - [warn] rpi-gpio : Raspberry Pi specific node set inactive
5 Jun 00:54:01 - [warn] rpi-gpio : Cannot find Pi RPi.GPIO python library
5 Jun 00:54:02 - [info] Settings file  : /data/settings.js
5 Jun 00:54:02 - [info] Context store  : 'default' [module=memory]
5 Jun 00:54:02 - [info] User directory : /data
5 Jun 00:54:02 - [warn] Projects disabled : editorTheme.projects.enabled=false
5 Jun 00:54:02 - [info] Flows file     : /data/flows.json
5 Jun 00:54:02 - [info] Creating new flow file
5 Jun 00:54:02 - [warn]
---------------------------------------------------------------------
Your flow credentials file is encrypted using a system-generated key.
If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
---------------------------------------------------------------------
5 Jun 00:54:02 - [info] Server now running at http://127.0.0.1:1880/
5 Jun 00:54:02 - [info] Starting flows
5 Jun 00:54:02 - [info] Started flows
```

Logs from Jun 4, 2020 to Jun 4, 2020 UTC

Demonstration 1: NodeRed Running Multiple Instances to Upgrade Software without Downtime

- 3 Replicas of NodeRed running



127.0.0.1:50977/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/

Overview

Workloads

- Cron Jobs
- Daemon Sets
- Deployments
- Jobs
- Pods
- Replica Sets
- Replication Controllers
- Stateful Sets

Discovery and Load Balancing

- Ingresses
- Services

Config and Storage

- Config Maps
- Persistent Volume Claims
- Secrets

Deployments

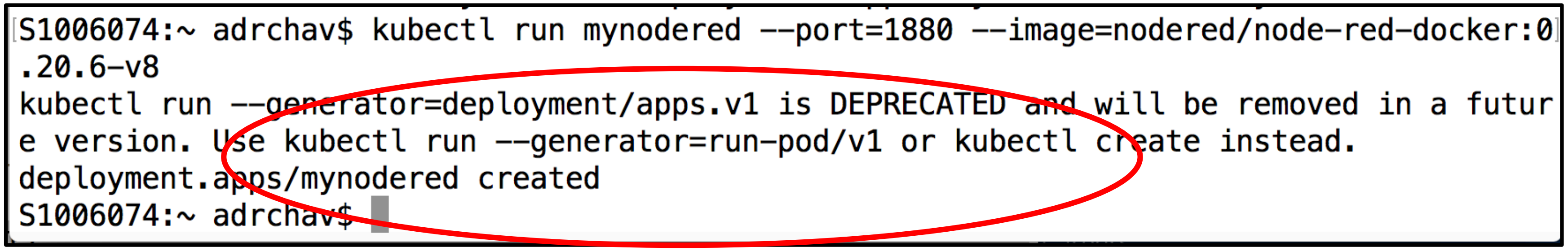
	Name	Namespace	Labels	Pods	Age ↑	Images
✓	mynodered	default	run: mynodered	3 / 3	20 minutes	nodered/node-red-docker:0.20.6-v8

1 – 1 of 1

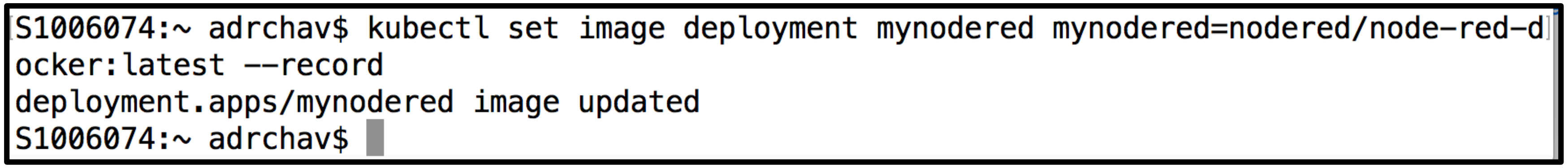
127.0.0.1:50977/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/#/daemonset?namespace=default

Demonstration 1: Upgrade NodeRed Software One Instance at a Time

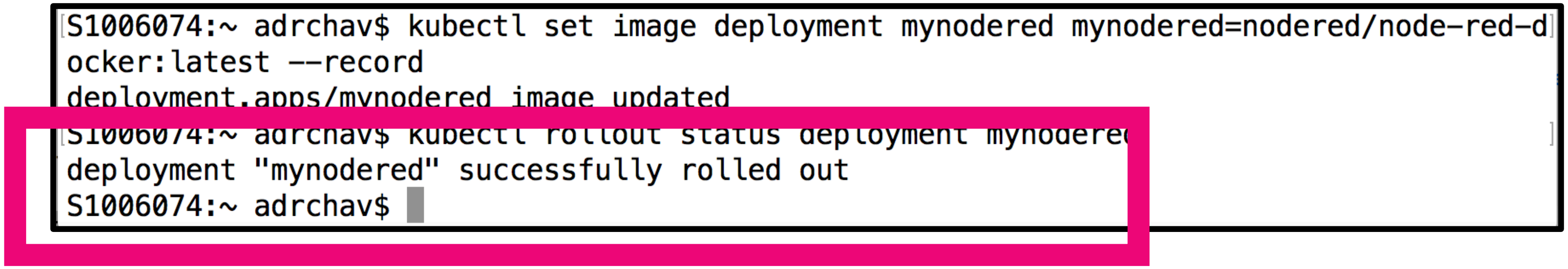
- Rolling update performed



```
S1006074:~ adrchav$ kubectl run mynodered --port=1880 --image=nodered/node-red-docker:0
.20.6-v8
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a futur
e version. Use kubectl run --generator=run-pod/v1 or kubectl create instead.
deployment.apps/mynodered created
S1006074:~ adrchav$
```



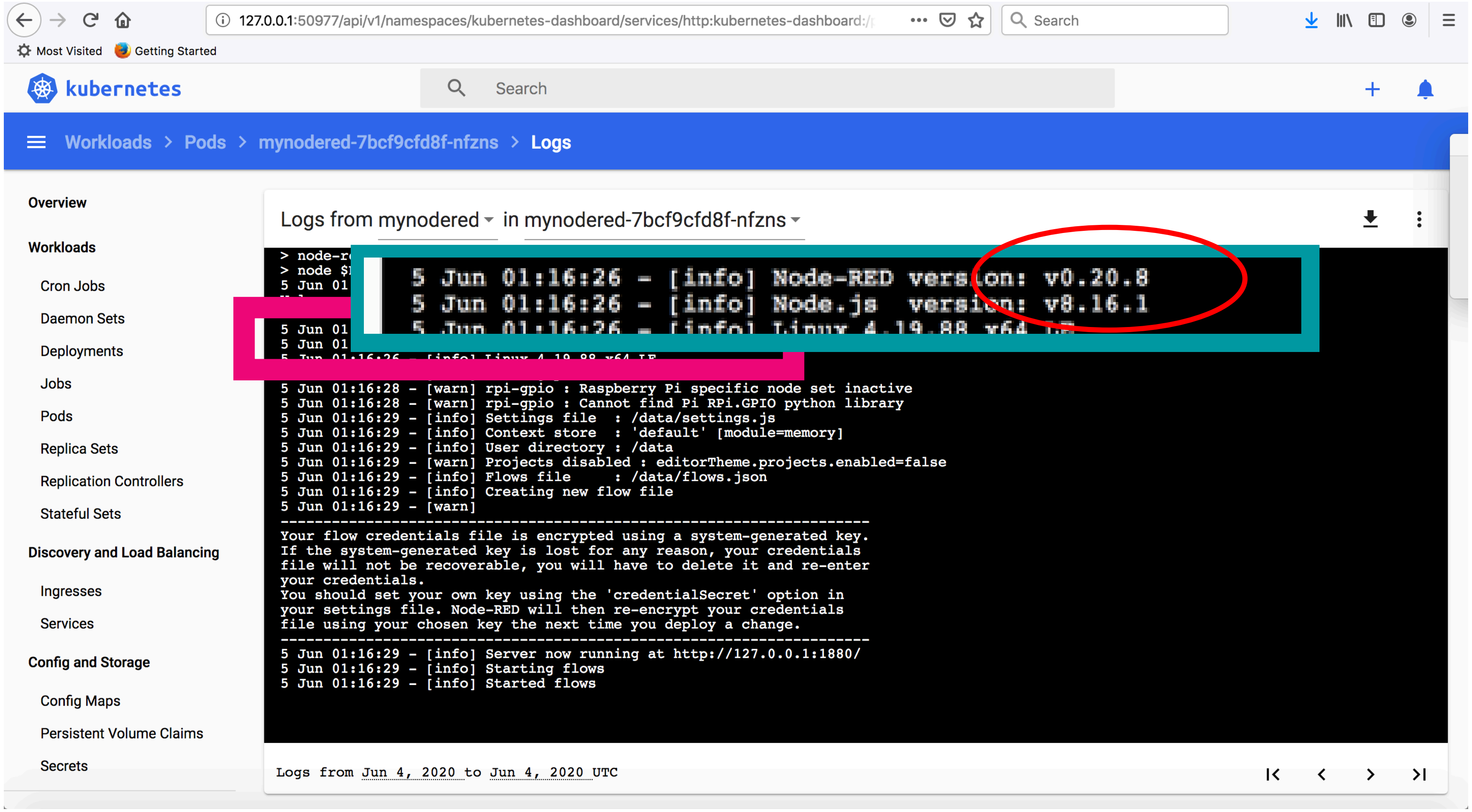
```
S1006074:~ adrchav$ kubectl set image deployment mynodered mynodered=nodered/node-red-d
ocker:latest --record
deployment.apps/mynodered image updated
S1006074:~ adrchav$
```



```
S1006074:~ adrchav$ kubectl set image deployment mynodered mynodered=nodered/node-red-d
ocker:latest --record
deployment.apps/mynodered image updated
S1006074:~ adrchav$ kubectl rollout status deployment mynodered
deployment "mynodered" successfully rolled out
S1006074:~ adrchav$
```

Demonstration 1: Upgrade NodeRed Software One Instance at a Time

- NodeRed = 0.20.8
- Node.js = 8.16.1



```
5 Jun 01:16:26 - [info] Node-RED version: v0.20.8
5 Jun 01:16:26 - [info] Node.js  version: v8.16.1
5 Jun 01:16:26 - [info] Linux 4.19.88 x64 LE
```

```
> node-r
> node $
5 Jun 01

5 Jun 01
5 Jun 01
5 Jun 01:16:26 - [info] Linux 4.19.88 x64 LE

5 Jun 01:16:28 - [warn] rpi-gpio : Raspberry Pi specific node set inactive
5 Jun 01:16:28 - [warn] rpi-gpio : Cannot find Pi RPi.GPIO python library
5 Jun 01:16:29 - [info] Settings file  : /data/settings.js
5 Jun 01:16:29 - [info] Context store  : 'default' [module=memory]
5 Jun 01:16:29 - [info] User directory : /data
5 Jun 01:16:29 - [warn] Projects disabled : editorTheme.projects.enabled=false
5 Jun 01:16:29 - [info] Flows file     : /data/flows.json
5 Jun 01:16:29 - [info] Creating new flow file
5 Jun 01:16:29 - [warn]
---------------------------------------------------------------------
Your flow credentials file is encrypted using a system-generated key.
If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
---------------------------------------------------------------------
5 Jun 01:16:29 - [info] Server now running at http://127.0.0.1:1880/
5 Jun 01:16:29 - [info] Starting flows
5 Jun 01:16:29 - [info] Started flows
```

Logs from Jun 4, 2020 to Jun 4, 2020 UTC

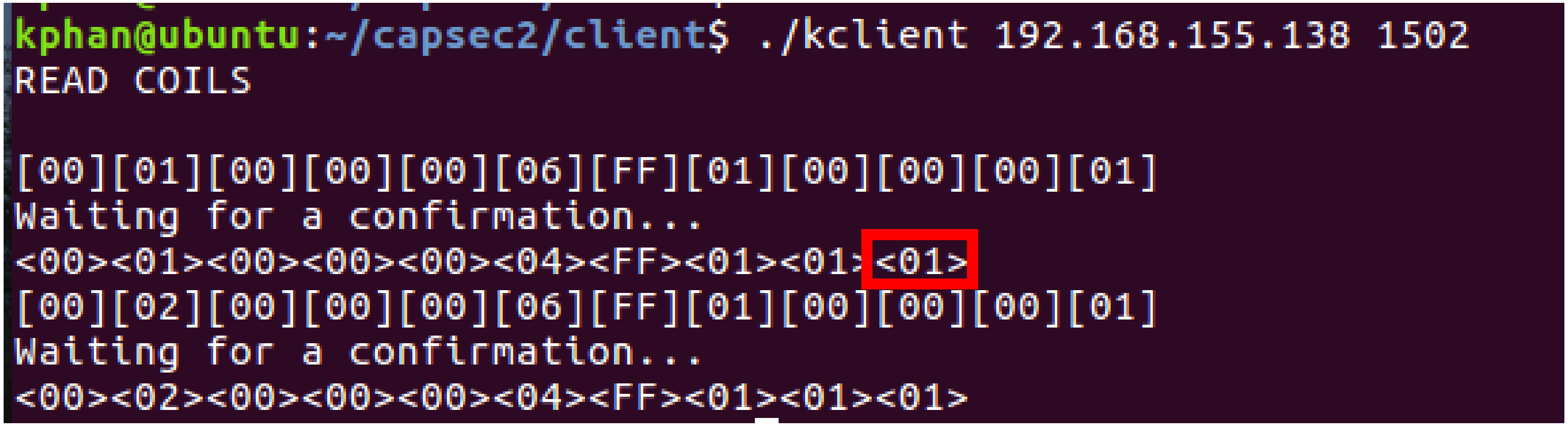
U.S. DEPARTMENT OF ENERGY | OFFICE OF Cybersecurity, Energy Security, and Emergency Response

Demonstration 2: Upgrading Stateful Programs Setup

- Two servers that will participate in upgrade
  - 1st server is the baseline and has an old unpatched version of the software running (kserver)
  - 2nd server has the updated software running that fixes the vulnerability (kserver2)
  - The software is using the libmodbus library to service Modbus queries
  - An exploit works against the baseline version on the 1st server and causes it to crash due to corrupting memory

Demonstration 2: Migration Process

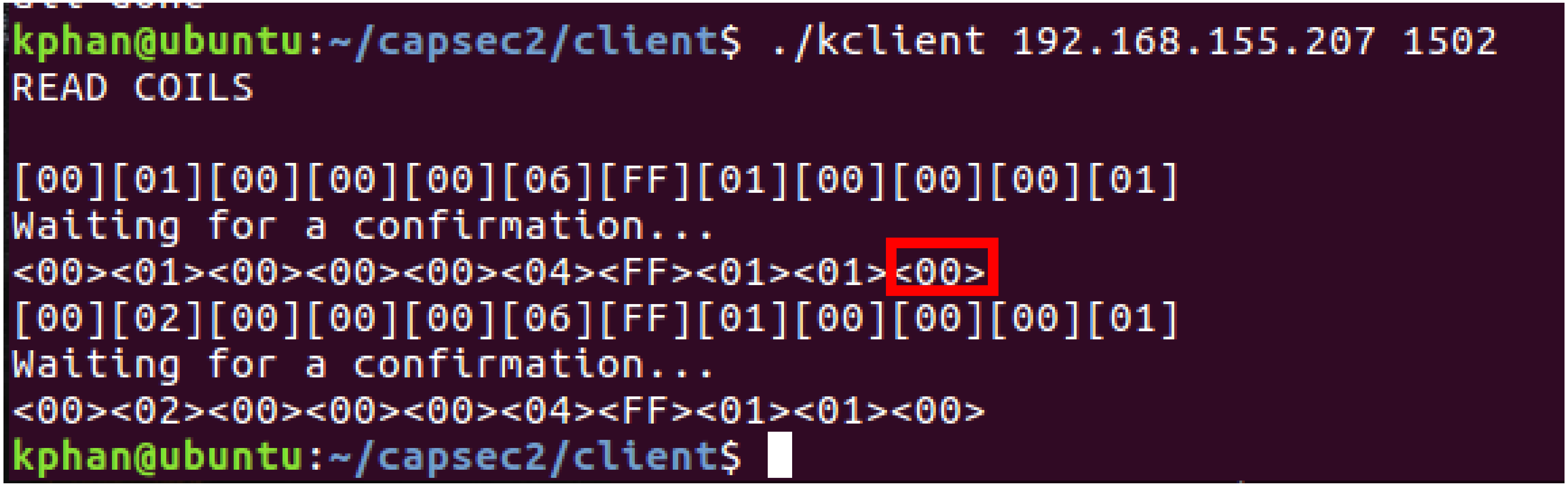
- Migrate agent running on server1 to pull out the state information from this server
- Migrate controller running on server2 that will wait for updates from the agent and will then update state information for server2
- Server1 has initial state value <01>



```
kphan@ubuntu:~/capsec2/client$ ./kclient 192.168.155.138 1502
READ COILS

[00][01][00][00][00][06][FF][01][00][00][00][01]
Waiting for a confirmation...
<00><01><00><00><00><04><FF><01><01><01>
[00][02][00][00][00][06][FF][01][00][00][00][01]
Waiting for a confirmation...
<00><02><00><00><00><04><FF><01><01><01>
```

Demonstration 2: Migration Process (cont.)

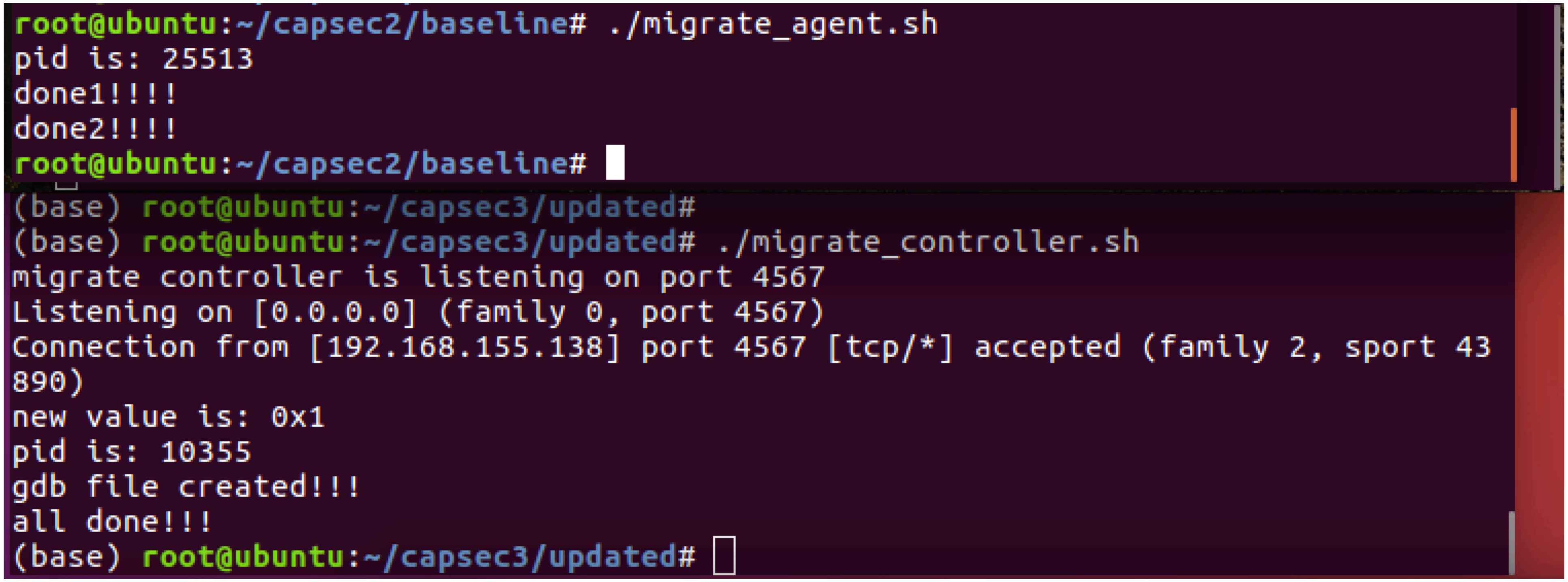
- Server1 has initial state value <00>



```
kphan@ubuntu:~/capsec2/client$ ./kclient 192.168.155.207 1502
READ COILS

[00][01][00][00][00][06][FF][01][00][00][00][01]
Waiting for a confirmation...
<00><01><00><00><00><04><FF><01><01><00>
[00][02][00][00][00][06][FF][01][00][00][00][01]
Waiting for a confirmation...
<00><02><00><00><00><04><FF><01><01><00>
kphan@ubuntu:~/capsec2/client$
```

Demonstration 2: Migration Software

- Top windows show the migration agent running on server1
- Bottom windows shows the migration controller – it has successful updated the state information of server2

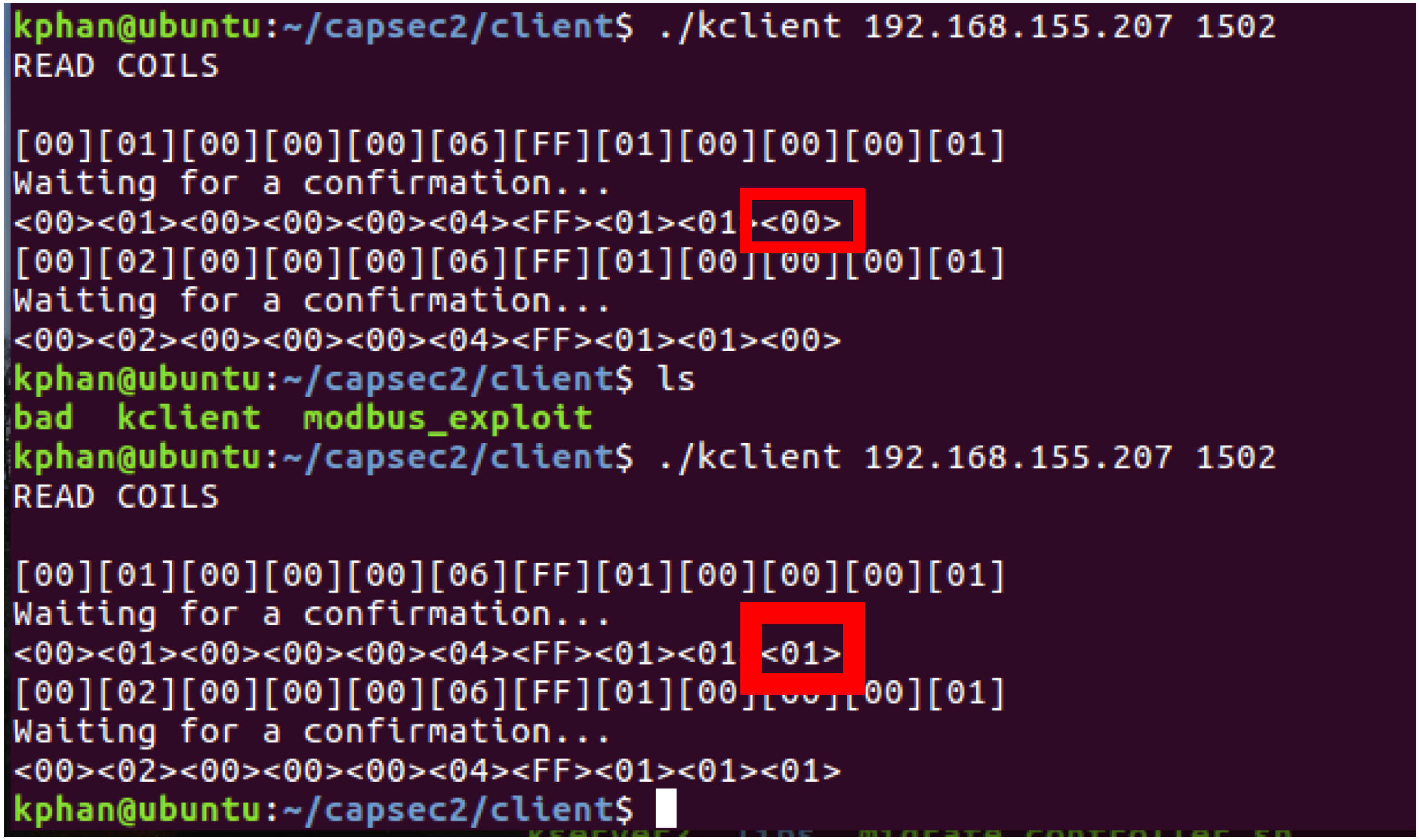


```
root@ubuntu:~/capsec2/baseline# ./migrate_agent.sh
pid is: 25513
done1!!!!
done2!!!!
root@ubuntu:~/capsec2/baseline# 
```

```
(base) root@ubuntu:~/capsec3/updated#
(base) root@ubuntu:~/capsec3/updated# ./migrate_controller.sh
migrate controller is listening on port 4567
Listening on [0.0.0.0] (family 0, port 4567)
Connection from [192.168.155.138] port 4567 [tcp/*] accepted (family 2, sport 43
890)
new value is: 0x1
pid is: 10355
gdb file created!!!
all done!!!
(base) root@ubuntu:~/capsec3/updated# 
```

Demonstration 2: Checking the State of server2 Again

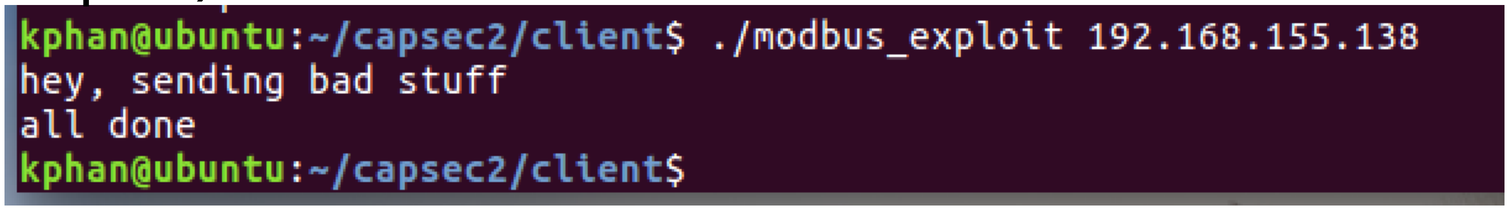
- After running the migration, checking the state of server2 shows the coil has been updated from <00> to <01>



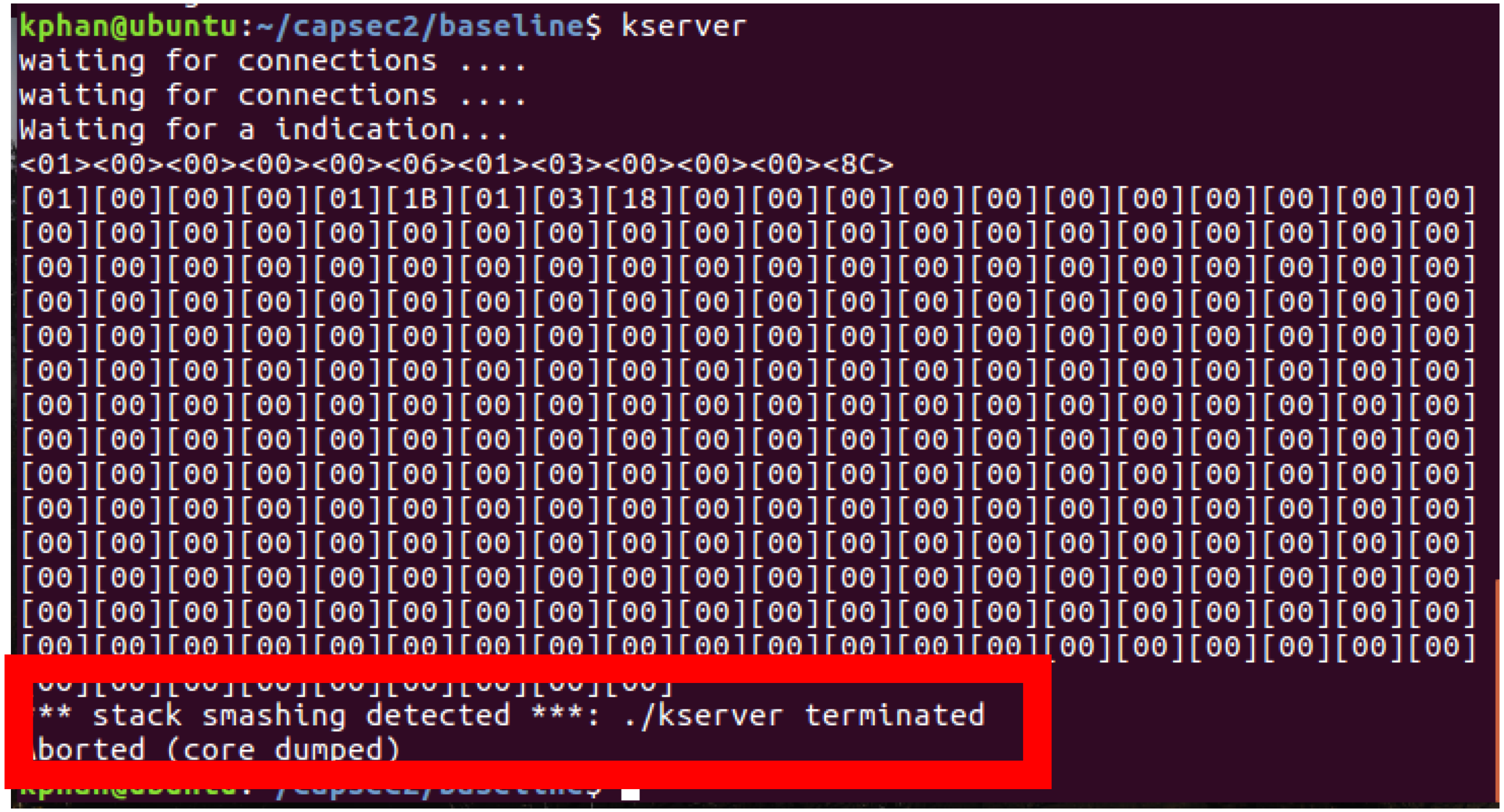
```
kphan@ubuntu:~/capsec2/client$ ./kclient 192.168.155.207 1502
READ COILS

[00][01][00][00][00][06][FF][01][00][00][00][01]
Waiting for a confirmation...
<00><01><00><00><00><04><FF><01><01><00>
[00][02][00][00][00][06][FF][01][00][00][00][01]
Waiting for a confirmation...
<00><02><00><00><00><04><FF><01><01><00>
kphan@ubuntu:~/capsec2/client$ ls
bad  kclient  modbus_exploit
kphan@ubuntu:~/capsec2/client$ ./kclient 192.168.155.207 1502
READ COILS

[00][01][00][00][00][06][FF][01][00][00][00][01]
Waiting for a confirmation...
<00><01><00><00><00><04><FF><01><01><01>
[00][02][00][00][00][06][FF][01][00][00][00][01]
Waiting for a confirmation...
<00><02><00><00><00><04><FF><01><01><01>
kphan@ubuntu:~/capsec2/client$
```

Demonstration 2: Running the Modus Exploit against server1

- Server1 crashes after exploit is launched (core dumped)



```
kphan@ubuntu:~/capsec2/client$ ./modbus_exploit 192.168.155.138
hey, sending bad stuff
all done
kphan@ubuntu:~/capsec2/client$
```

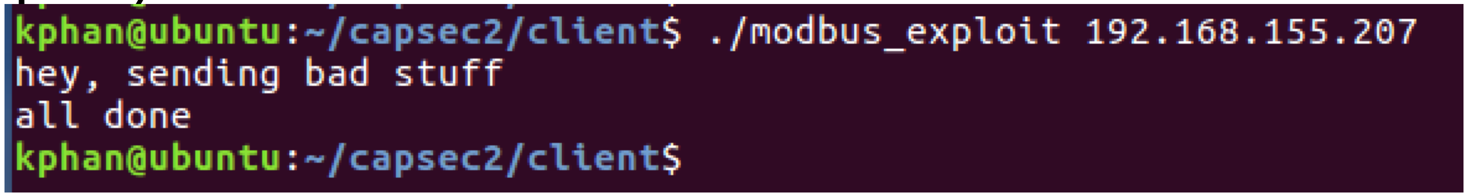


```
kphan@ubuntu:~/capsec2/baseline$ kserver
waiting for connections ....
waiting for connections ....
Waiting for a indication...
<01><00><00><00><00><06><01><03><00><00><00><8C>
[01][00][00][00][01][1B][01][03][18][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]
00][00][00][00][00][00][00][00][00]
** stack smashing detected ***: ./kserver terminated
borted (core dumped)
kphan@ubuntu:~/capsec2/baseline$
```

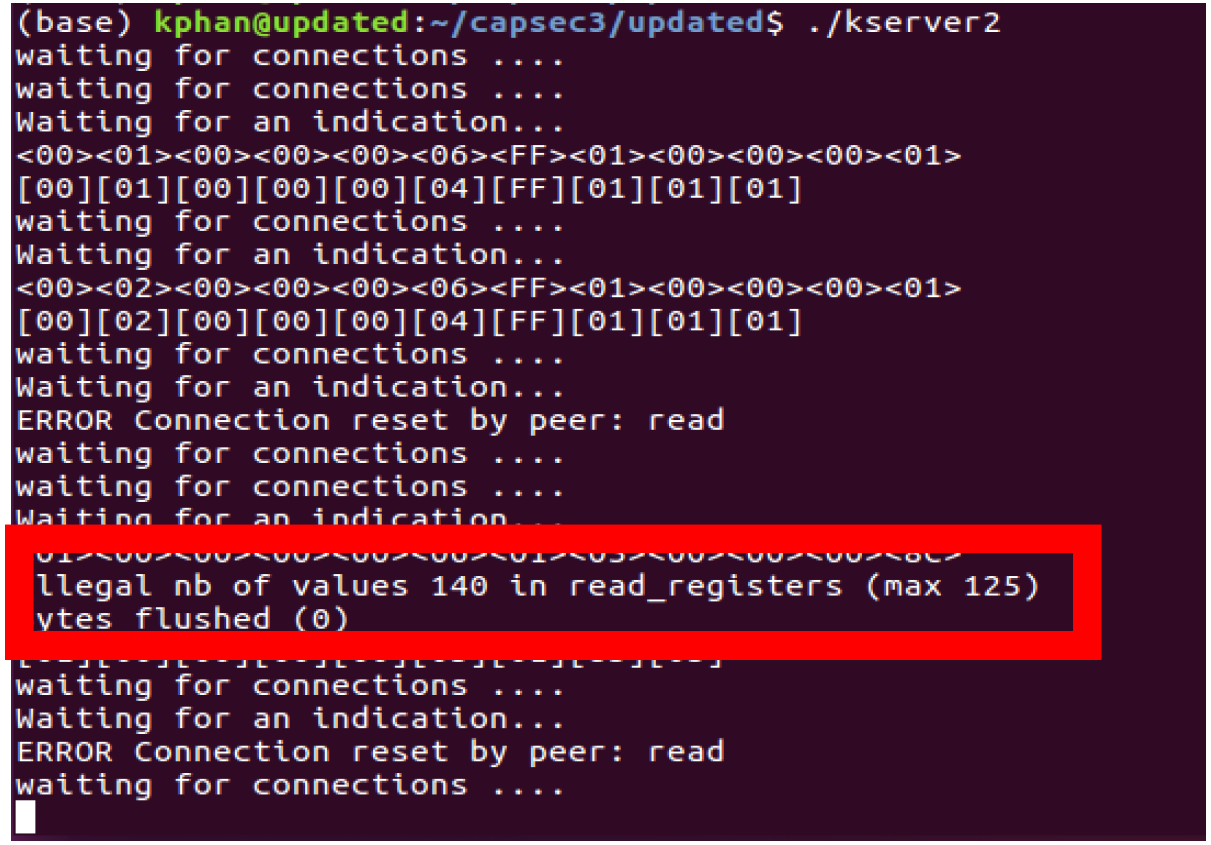
U.S. DEPARTMENT OF ENERGY | OFFICE OF Cybersecurity, Energy Security, and Emergency Response

Demonstration 2: Running the Modus Exploit against server2

- Server2 continues to run after exploit is launched (NO core dumped)



```
kphan@ubuntu:~/capsec2/client$ ./modbus_exploit 192.168.155.207
hey, sending bad stuff
all done
kphan@ubuntu:~/capsec2/client$
```



```
(base) kphan@updated:~/capsec3/updated$ ./kserver2
waiting for connections ....
waiting for connections ....
Waiting for an indication...
<00><01><00><00><00><06><FF><01><00><00><00><01>
[00][01][00][00][00][04][FF][01][01][01]
waiting for connections ....
Waiting for an indication...
<00><02><00><00><00><06><FF><01><00><00><00><01>
[00][02][00][00][00][04][FF][01][01][01]
waiting for connections ....
Waiting for an indication...
ERROR Connection reset by peer: read
waiting for connections ....
waiting for connections ....
Waiting for an indication...
01><00><00><00><00><00><01><03><00><00><00><8C>
llegal nb of values 140 in read_registers (max 125)
ytes flushed (0)
waiting for connections ....
Waiting for an indication...
ERROR Connection reset by peer: read
waiting for connections ....
```