Module 12 Lab: Linux Automation via “Attack of the Clones”

> 💡This week we will explore more advanced Desktop virtualization concepts while exploring how systems administration can be conducted at scale using automation tools such as Ansible. You have three new Linux VMs, where one will be used to control the other two through various means.

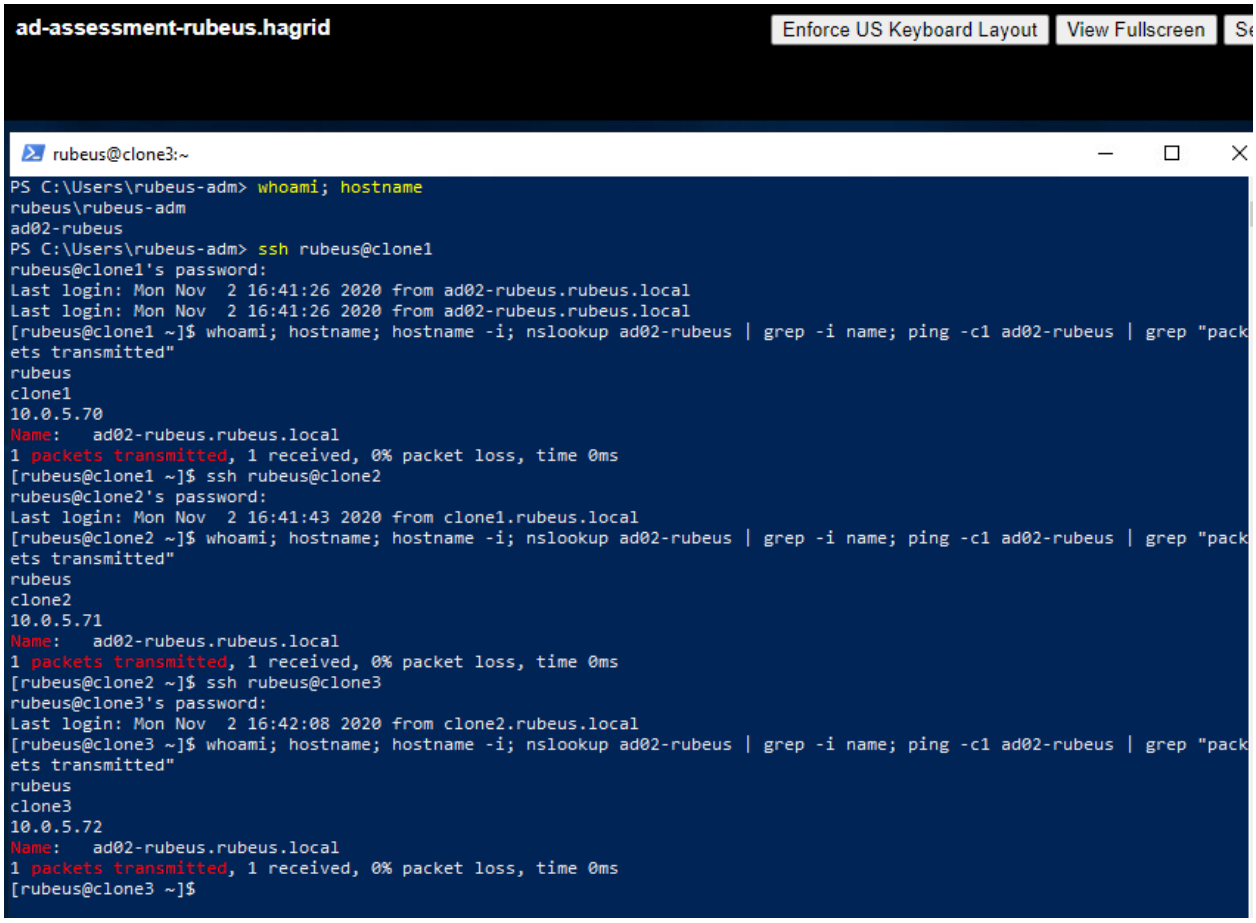
Initial Setup

You have 3 new Linux hosts in your environment, so let’s configure them similarly as you’ve done with previous new Linux hosts. For each system, make your named sudo user with the exact same name and password on each. Also, these are stand-alone hosts, and thus do not need to be joined to your AD domain.

- 10.0.5.70 (hostname: clone1)
- 10.0.5.71 (hostname: clone2)
- 10.0.5.72 (hostname: clone3)

Starting on AD, let’s do some testing via ssh into the clone1 to affirm connectivity and resolution across the clones.

`Deliverable 1. Perform multiple routine testing for connectivity and name resolution via 1-liners similar to the below screen:`



PSSH

Install both of the following yum packages on clone01 which assume ‘yes’ to any question prompted during installation:

- epel-release
- pssh

We are going to use a different authentication technique for SSH. We will create an RSA public and private key-pair, with the private key protected by a *passphrase*. Make sure to use the default key names (id_rsa.pub and id_rsa).

Select rubeus@clone1:~

```
[rubeus@clone1 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/rubeus/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/rubeus/.ssh/id_rsa.
Your public key has been saved in /home/rubeus/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:z/WNuJDh44DJTiNoxEjAhCjxD659oEBMqv2xdrRPv5E rubeus@clone1
The key's randomart image is:
+---[RSA 2048]----+
|B+               |
|Oo               |
|o+o              |
|+= o             |
|+ * o . S . .    |
|.= + = + + = o o |
|o + * O o E . o .|
| . o = + + + .   |
|      . . +..    |
+----[SHA256]-----+
[rubeus@clone1 ~]$
```

We are going to push the public component of this keypair (id_rsa.pub) to our accounts on clone2 and clone3.

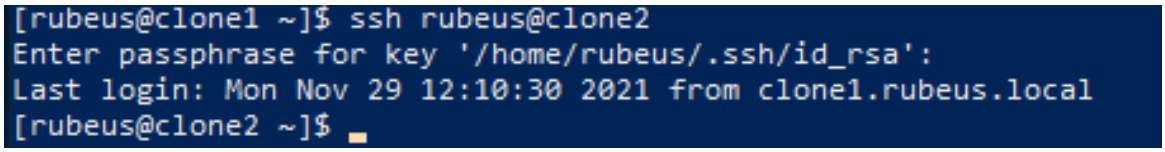
```
[rubeus@clone1 ~]$ ls -l .ssh/
total 12
-rw-------. 1 rubeus rubeus 1766 Nov 29 11:41 id_rsa
-rw-r--r--. 1 rubeus rubeus  395 Nov 29 11:41 id_rsa.pub
-rw-r--r--. 1 rubeus rubeus  178 Nov 29 11:31 known_hosts
[rubeus@clone1 ~]$ ssh-copy-id rubeus@clone2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/rubeus/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
rubeus@clone2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'rubeus@clone2'"
and check to make sure that only the key(s) you wanted were added.

[rubeus@clone1 ~]$
```

When a public key is copied via ssh-copy-id, it shows up on the remote system here:

```
[rubeus@clone2 ~]$ cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCwj91KuRV377/D1Z3g0bFva8wKCJ5MDNvqTjgg+tBQZLp96gq+HroU+ZLI5QmbR6qGvNYfaij+U0/wwWA
IEAURFDK/C8/AIBwU0UAkjY9EM9UKWzDApC/X9c9yN84XmZec0whpkVSKfJIHJzOcEu16OWQSkCJfL4q1DK3UrqcEOoMUN6VbA4wdtlVvs+yob005cRBQTC
JBQa+rew8CbCqr2L4fQrhC+tq4Lx6c8VSGZ7188D8+3MMn0ukejPsXUusmB9sV+lwPsd9IeZtb1pyIA0ngDXC9khDYXatjwgiuZEAma66kFRs9/+jJf0QdK
pSYHMuGs/4dOwhDyrBPx rubeus@clone1
[rubeus@clone2 ~]$
```

💣Although protected by a passphrase, the private key if revealed will compromise the security of the associated account. This becomes particularly important when that account is used for multiple servers. The private key should **always** be protected!

Deliverable 2. ssh into either clone2 or clone3 using your ssh key. The passphrase you enter is only for unlocking your local private key on clone1, as opposed to logging into the remote system itself. Provide a screenshot that shows the prompt for your passphrase as well as the login into clone2 or 3 that does not ask for a password.

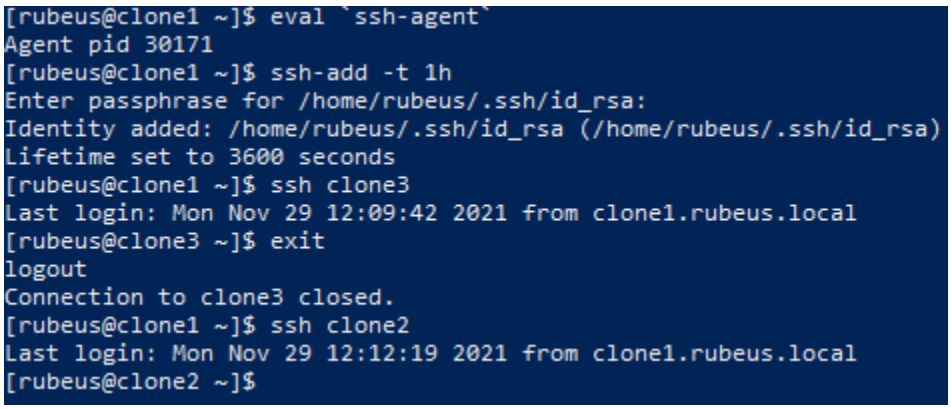


ssh-agent

Far too many administrators create ssh keys that are not protected by a passphrase. This is analogous to leaving the keys to your Porsche laying around. They do this because they still need to type in a passphrase to unlock the keys if they are so protected. We will balance the security provided with a passphrase against the convenience of a totally passwordless solution by "caching" the passphrase in memory for an hour using the ssh-agent program.

The following screenshot shows how to load the ssh-agent if it is not already loaded (note the rarely used back-ticks above Tab) followed by adding the private key for 1 hour. The subsequent ssh into clone3 does not prompt for a password. Logout of clone3 with exit and login to clone2.

Deliverable 3. Provide a screenshot showing passwordless login to clone2 or 3 after having loaded the ssh-agent and private key.



/etc/sudoers

On both clone2 and clone3, adjust /etc/sudoers so that the highlighted line is uncommented:

```
  GNU nano 2.3.1                File: /etc/sudoers

## The COMMANDS section may have other options added to it
##
## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL

## Allows members of the 'sys' group to run networking, so
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELE

## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)       ALL

## Same thing without a password
%wheel  ALL=(ALL)       NOPASSWD: ALL
```

This allows elevation to root without retyping a password if the current user is in the wheel group and is a common configuration.

Deliverable 4. Provide a screenshot similar to the one below that shows passwordless access to clone2 or clone3 and elevation to root without retyping a password.

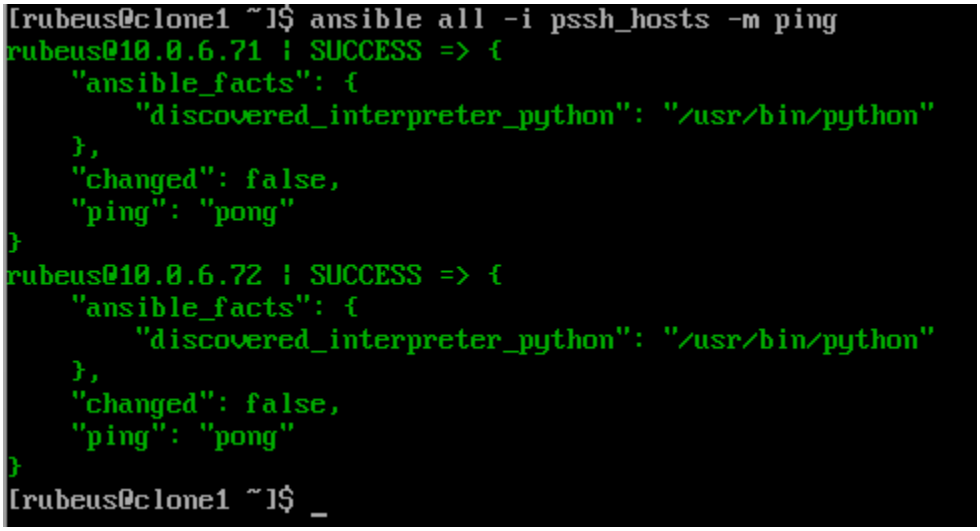
```
[rubeus@clone1 ~]$ ssh clone2
Last login: Mon Nov 29 12:35:14 2021 from clone2.rubeus.local
[rubeus@clone2 ~]$ sudo -i
[root@clone2 ~]#
```

Deliverable 5. Review the man page for pssh and construct a pssh hosts file containing your clone2 and clone3. Then execute the following non-privileged and privileged commands displaying inline standard output & errors as each host completes. Provide screenshots showing the command and [SUCCESS OUTPUT] for all four commands:

- uptime
- uname -a
- sudo yum -y install tree
- tree /etc/yum.repos.d/

Ansible

Deliverable 6. Install the ansible package using yum on just clone1. Once installed, conduct the following test that walks through all hosts in your hosts file and runs a module called ping against them. Take a screenshot similar to the one below that shows a ping and pong response from clone2 and clone3.



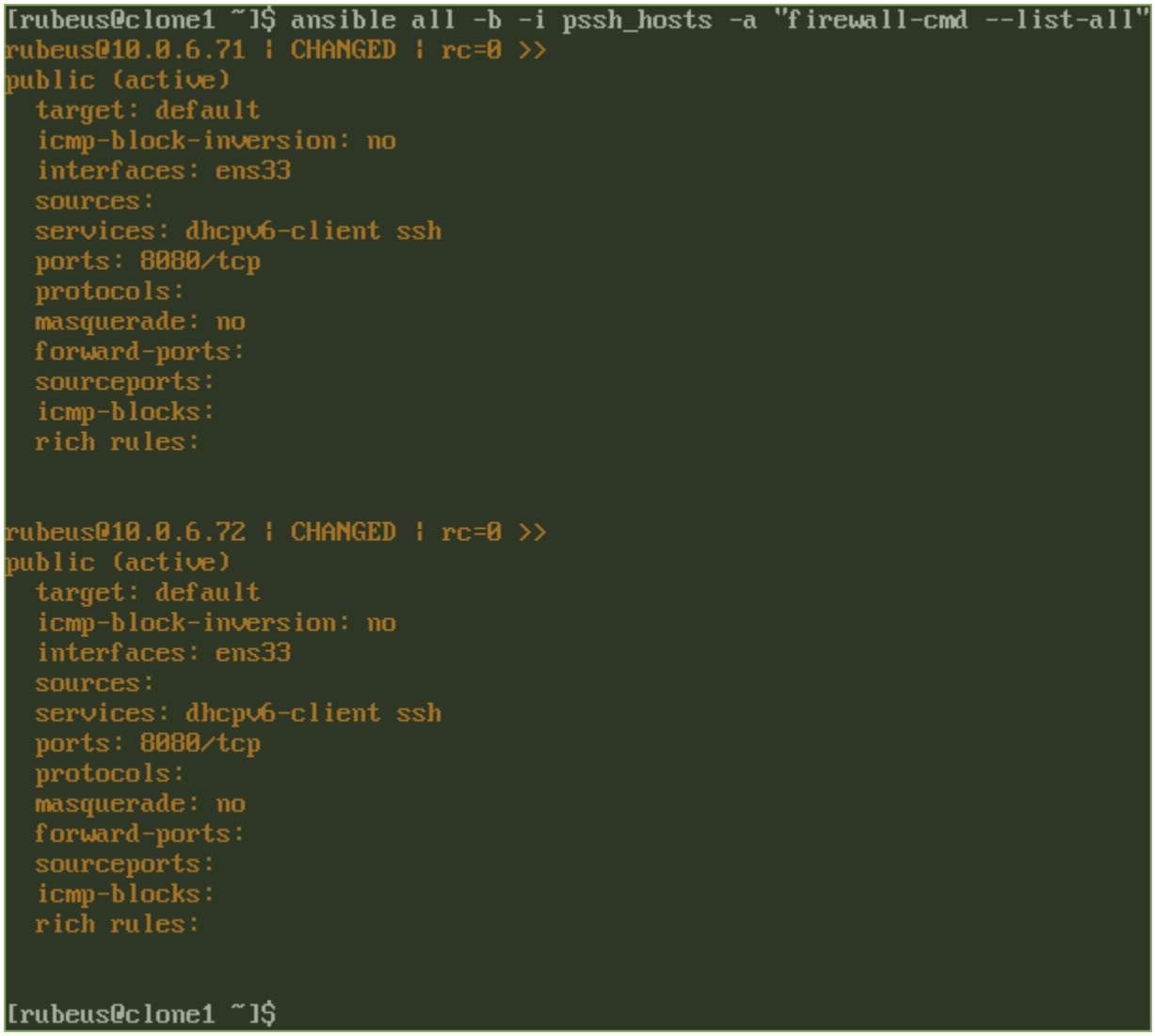
Ansible and sudo

Consider the following screenshot. As you should recall, the /etc/passwd file is readable by everyone and the /etc/shadow file (which contains hashed passwords) is only readable by root. Notice the success on tailing the /etc/passwd file and subsequent failure on /etc/shadow. This is resolved by telling Ansible that the user associated with the ssh public key at the other end of the connection is a sudoer user (-b).

Deliverable 7. Provide a screenshot similar to the one below.

```
[rubeus@clone1 ~]$ ansible all -i pssh_hosts -a "tail -n 1 /etc/passwd"
rubeus@10.0.6.72 | CHANGED | rc=0 >>
rubeus:x:1000:1000::/home/rubeus:/bin/bash

rubeus@10.0.6.71 | CHANGED | rc=0 >>
rubeus:x:1000:1000::/home/rubeus:/bin/bash

[rubeus@clone1 ~]$ ansible all -i pssh_hosts -a "tail -n 1 /etc/shadow"
rubeus@10.0.6.71 | FAILED | rc=1 >>
tail: cannot open ‘/etc/shadow’ for reading: Permission deniednon-zero return code

rubeus@10.0.6.72 | FAILED | rc=1 >>
tail: cannot open ‘/etc/shadow’ for reading: Permission deniednon-zero return code

[rubeus@clone1 ~]$ ansible all -b -i pssh_hosts -a "tail -n 1 /etc/shadow"
rubeus@10.0.6.71 | CHANGED | rc=0 >>
rubeus:$6$PGNBwi2t$GmxO7e.PzidsbdX6IUNDy0E7UyVPxop494NcwIufdlYBa93ref14ExNP9A7o1JJQuLQtH3QxkI6kNlOZa
borB/:18216:0:99999:7:::

rubeus@10.0.6.72 | CHANGED | rc=0 >>
rubeus:$6$CB4iMmSK$zpoXjLoXB3dcQJD7w3QYpxGqSZ4ARjq/Gonqqlv0sA3XzAGrsZbhyyz06Tj6UPcL/xrPPJO28fuBRtvDd
6SJc.:18216:0:99999:7:::

[rubeus@clone1 ~]$ _
```

Deliverable 8. Figure out how to add an arbitrary port to the firewall using Ansible. Show the commands used, their success, and then issue another command to list those ports as shown in the following screenshot (8080/tcp in the example).



The Ansible Playbook

On clone01, create a directory called “nginx”.

Within that directory, wget the following file: wget https://gist.githubusercontent.com/icasimpan/803955c7d43e847ce12ff9422c1cbbc4/raw/c1753594e638590ac4d54e685dd3ae1ee1d9f40a/nginx-centos7.yml

(If you are typing the link by hand, you might consider PuTTY from your CYBER.LOCAL workstation).

Make the SYS-255 modifications shown below to that file:

```
## Credits to John Lieske - https://www.ansible.com/blog/getting-started-writing-your-first-playbook
---
- name: Install nginx
#SYS255 Modification to hosts
  hosts: all
  become: true

  tasks:
  - name: Add epel-release repo
    yum:
      name: epel-release
      state: present

  - name: Install nginx
    yum:
      name: nginx
      state: present

  - name: Insert Index Page
    template:
#SYS255 - Create and cusomize this file in your local directory on clone1
      src: index.html
      dest: /usr/share/nginx/html/index.html

  - name: Start NGiNX
    service:
      name: nginx
      state: started

#SYS255 Modification  add a firewall rule and reload the firewall
  - name: Enable Firewall
    command: firewall-cmd --add-service=http --permanent

  - name: Reload Firewall
    command: firewall-cmd --reload
~
```

* Note: the trailing '-' in the screenshot above is shown in vi, not the actual file.

Running the Playbook

Consider the following screenshot. Your account's 1-hour of ssh-agent time has expired, so the private key needed to be reloaded. After executing the script, the curl command was used to validate that nginx was serving the index.html that was copied from the local directory.

Deliverable 9. Provide a screenshot that shows a successful playbook run followed by curls to the index page for clone2 and clone3.

```
[rubeus@clone1 nginx]$ eval `ssh-agent`
Agent pid 2933
[rubeus@clone1 nginx]$ ssh-add
Enter passphrase for /home/rubeus/.ssh/id_rsa:
Identity added: /home/rubeus/.ssh/id_rsa (/home/rubeus/.ssh/id_rsa)
[rubeus@clone1 nginx]$ ansible-playbook nginx-centos7.yml -i ../pssh_hosts _
```

```
TASK [Add epel-release repo] ***************************************************
changed: [rubeus@10.0.6.72]
changed: [rubeus@10.0.6.71]

TASK [Install nginx] ***********************************************************
changed: [rubeus@10.0.6.71]
changed: [rubeus@10.0.6.72]

TASK [Insert Index Page] *******************************************************
changed: [rubeus@10.0.6.72]
changed: [rubeus@10.0.6.71]

TASK [Start NGiNX] *************************************************************
changed: [rubeus@10.0.6.72]
changed: [rubeus@10.0.6.71]

TASK [Enable Firewall] *********************************************************
changed: [rubeus@10.0.6.72]
changed: [rubeus@10.0.6.71]

TASK [Reload Firewall] *********************************************************
changed: [rubeus@10.0.6.71]
changed: [rubeus@10.0.6.72]

PLAY RECAP *********************************************************************
rubeus@10.0.6.71           : ok=7    changed=6    unreachable=0    failed=0    skipped=0    rescued=
0    ignored=0
rubeus@10.0.6.72           : ok=7    changed=6    unreachable=0    failed=0    skipped=0    rescued=
0    ignored=0

[rubeus@clone1 nginx]$ curl clone2
howdy sys255
[rubeus@clone1 nginx]$ curl clone3
howdy sys255
```