GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials (Security 401)"
at http://www.giac.org/registration/gsec

Introduction to Rundeck for Secure Script Executions

GIAC (GSEC) Gold Certification

Author: John Becker, jbecker42@gmail.com
Advisor: Stephen Northcutt
Accepted: August 10, 2016

Abstract

Many organizations today support physical, virtual, and cloud-based systems across a wide range of operating systems. Providing least privilege access to systems can be a complex mesh of sudoers files, profiles, policies, and firewall rules. While configuration management tools such as Puppet or Chef help ensure consistency, they do not inherently simplify the process for users or administrators. Additionally, current DevOps teams are pushing changes faster than ever. Keeping pace with new services and applications often force sysadmins to use more general access rules and thus expose broader access than necessary. Rundeck is a web-based orchestration platform with powerful ACLs and ssh-based connectivity to a wide range of operating systems and devices. The simple user interface for Rundeck couples with DevOps-friendly REST APIs and YAML or XML configuration files. Using Rundeck for server access improves security while keeping pace with rapidly changing environments.

© 2016 The SANS Institute Author retains full rights.

1. Introduction

Simplicity and least privilege access are two pillars of information security. A complex system introduces more defects and is hard to test (Schneier, 1999). NIST guidelines state "…organizations should base access control policy on the principle of least privilege…" (Swanson & Guttman, 1996). However, simplicity and least privilege can be at odds with each other in an organization supporting a disparate collection of operating systems, network devices, and hybrid cloud environments. Each operating system for these devices has a different access control implementation. Unix and Linux have sudo and SELinux, Windows has Group Policies, and network devices have various options. This system complexity combined with elastic cloud environments and DevOps change velocity results in conflict with security policies and access controls (Lawler, 2015).

Least privilege access controls are often implemented with Role-Based Access Controls (RBAC) but are "often difficult or costly to achieve because it is difficult to tailor access based on various attributes or constraints" (Hu, Ferraiolo & Kuhn, 2006). It is difficult for a single system to keep pace with automated configuration management and dynamic virtual environments. There are still times when a user needs an interactive session (e.g. SSH) to complete a task. However, the majority of commands can be processed centrally, either interactively or scheduled, and outlier requests can be flagged and granted temporary access. Although many tools enable configuration management and remote access, Rundeck stands out as a robust solution for controlling access by combining a user-friendly interface and granular access controls.

Rundeck is an open source orchestration, job scheduler, and runbook automation platform written in Java (rundeck.org, 2016). The same tools and scripts DevOps, SysAdmin, and Security teams use today can be implemented with granular access controls in Rundeck. In turn, these scripts and tools can be safely delegated to different groups in a controlled manner. This delegation of access can be version-controlled and scripted to keep pace with DevOps teams. These scripted executions are logged centrally in the Rundeck database. Operations and development teams maintain their release

© 2016 The SANS Institute Author retains full rights.

velocity while security teams see least privilege access, improved logging, and a shared tool for incident response.

2. Rundeck Basics

Some refer to Rundeck as "the Swiss army knife for ops" (rundeck.org, 2016). The application's principle function is to execute jobs (scripts or commands) on target nodes. The web user interface, command line, API, and scheduler are all job triggers. Jobs can be bundled together for tasks such as automated runbooks, software deployments, and incident response.

2.1. Setup

Rundeck is available as a Java jar-based "launcher" install, Debian or Ubuntu package, or an RPM package. The examples in this document will use Ubuntu 16.04. Begin with installing Java:

```
sudo apt-get install openjdk-7-jdk
```

Then download the latest Rundeck Debian package and verify the SHA hash from http://rundeck.org/downloads.html.

```
wget http://dl.bintray.com/rundeck/rundeck-deb/rundeck-2.6.8-1-GA.deb

shasum rundeck-2.6.8-1-GA.deb
865c669c8694a9b6fa595363c9906cf771818337 rundeck-2.6.8-1-GA.deb
```

Check the SHA hash matches what is posted on rundeck.org, then install the package and start the rundeckd service.

```
sudo dpkg -i rundeck-2.6.8-1-GA.deb
sudo service rundeckd start
```

By default, Rundeck runs as a non-privileged user "rundeck" on 0.0.0.0 (all interfaces) on TCP port 4440. Users can login as "admin" for the default username and password at http://<hostname>:4440. The built-in HSQLDB database suffices for small instances, but most production use cases need a dedicated relational database (Rundeck

© 2016 The SANS Institute Author retains full rights.

Administrator Guide, 2016). MySQL is the preferred database with support for MS-SQL as well (Schueler, 2015). Scaling Rundeck is outside the scope of this paper.

2.2. User Authentication

Rundeck supports three types of authentication: PropertyFileLoginModule (/etc/rundeck/realm.properties), LDAP, and PAM. Many organizations prefer LDAP for centralized access. The PAM module works well in settings where users are managed locally with configuration management (e.g. Puppet). PropertyFileLoginModule is the default and sufficient for test instances of Rundeck. The PropertyFileLoginModule supports three types of hashing or obfuscation for passwords: OBF, MD5, and CRYPT. These are built-in to the Jetty project used by Rundeck.

Of the supported types, OBF is the least secure as it is a reversible obfuscation method (Jetty, 2016). MD5 has been insecure for password hashing for nearly 20 years (Dobbertin, 1996). CRYPT is the UnixCrypt Java Class (Jetty Source Code, 2016) that is limited to a 56bit DES algorithm (Class UnixCrypt, 2016). Numerous superior tools exist for encrypting, auditing, and managing LDAP and PAMPropertyFileLoginModule. Therefore, LDAP and PAM are the recommended production authentication methods.

2.3. Hardening

A tool like Rundeck has keys to access a broad range of critical systems in an organization. Rundeck servers should be placed in a protected enclave because of the need for confidentiality and integrity (Rome, n.d.). Inbound connections are limited to HTTPS for users and SSH for Rundeck administration. Outbound connections depend on the node plugins used, but typically require SSH on port 22 at a minimum. After securing network access to the Rundeck, care should be taken to configure and monitor the host itself.

2.3.1. File System

Rundeck installs with file permissions for the directory /etc/rundeck set to 655 and owned by root. Files inside of /etc/rundeck have ownership of "rundeck" and group "rundeck" with 640 file permissions. These items are sensitive as they comprise Access Control Lists (*.aclpolicy), user accounts (realm.properties), and core configuration files that include passwords (*.properties). From an operating system perspective, this may be

© 2016 The SANS Institute Author retains full rights.

acceptable as the “rundeck" user has no shell access. However, the Rundeck application has a loophole wherein users can execute jobs locally as the “rundeck" user in Linux. This access allows for non-privileged Rundeck users to read and modify critical items in /etc/rundeck when executing jobs or ad-hoc commands against the “localhost" node. Restricting "localhost" targets to Rundeck administrators prevents inadvertent access to critical configurations. An additional layer of protection is to change the file ownership inside /etc/rundeck to “root:rundeck". The 640 file permissions will allow the "rundeck" user read access, but limit writes to the root user. Finally, files in /etc/rundeck should be monitored closely for any changes with a Host IDS program such as Tripwire or OSSEC.

```
root@galactica:/etc/rundeck# chown root:rundeck *
root@galactica:/etc/rundeck# ls -al
total 56
drwxr-xr-x   3 root root    4096 Jul 17 21:00 .
drwxr-xr-x 102 root root    4096 Jul 17 21:17 ..
-rw-r-----   1 root rundeck  738 Jun 10 13:37 admin.aclpolicy
-rw-r-----   1 root rundeck 1104 Jun 10 13:37 apitoken.aclpolicy
-rw-r-----   1 root rundeck  511 Jun 10 13:37 cli-log4j.properties
-rw-r-----   1 root rundeck 1284 Jun 10 13:37 framework.properties
-rw-r-----   1 root rundeck  141 Jun 10 13:37 jaas-loginmodule.conf
-rw-r-----   1 root rundeck 7661 Jun 10 13:37 log4j.properties
-rw-r-----   1 root rundeck 1788 Jun 10 13:37 profile
-rw-r-----   1 root rundeck  549 Jun 10 13:37 project.properties
-rw-r-----   1 root rundeck  986 Jun 10 13:37 realm.properties
-rw-r-----   1 root rundeck  416 Jun 10 13:37 rundeck-config.properties
drwxr-xr-x   2 root rundeck 4096 Jul 17 21:00 ssl
```

2.3.2. Remove Default Access

The first step for a production Rundeck instance setup should be to replace the admin user login per CSC 5.3 (Critical Security Controls, 2015). The default account resides in the last section of /etc/rundeck/realm.properties. Delete or comment out the section below if using LDAP or PAM. If forced to use the PropertyFileLoginModule authentication, change the ‘admin' username and password and store in MD5 format per the instructions at http://rundeck.org/docs/administration/authenticating-users.html.

© 2016 The SANS Institute Author retains full rights.

```
# This sets the default user accounts for the Rundeck app
#
admin:admin,user,admin,architect,deploy,build
```

2.3.3. Enabling Security Transport

Rundeck listens in cleartext at http://<hostname>:4440. Cleartext HTTP transport is a bad practice for system administration. CSC 3.4 states strong encryption (TLS) should be used (Critical Security Controls, 2015). There are two options for providing HTTPS transport security. The first choice is to enable SSL per the instructions at http://rundeck.org/docs/administration/configuring-ssl.html. This configuration works and results in the Rundeck service listening at https://<hostname>:4443. However, there is another option to use a web proxy for SSL termination in front of Rundeck. Apache has the added benefits of listening on the standard HTTPS port of 443, enabling support for Multi-Factor Authentication, as well as providing options for additional logging and web application firewalls.

The example below will focus on Apache for SSL termination, reverse-proxy, and Google Authenticator. The first step is to install Apache and enable the required proxy and SSL modules:

```
sudo apt-get install apache2
sudo a2enmod proxy
sudo a2enmod proxy_http
sudo a2enmod rewrite
sudo a2enmod deflate
sudo a2enmod headers
sudo a2enmod proxy_connect
sudo a2enmod proxy_html
sudo a2enmod ssl
```

Next, configure the SSL certificate and private key used by Apache in /etc/ssl/certs and /etc/ssl/private. A standard x.509 SSL certificate and key is available from providers such as Verisign or Microsoft Certificate Services. For this paper, a private certificate will be used:

© 2016 The SANS Institute Author retains full rights.

```
sudo openssl req -x509 -nodes -days 365 -newkey
rsa:2048 -keyout /etc/ssl/private/rundeck-apache.key -out
/etc/ssl/certs/rundeck-apache.crt
```

Once the certificate and key are ready, apply the configurations below for Apache to redirect (mod_proxy) and terminate SSL connections (mod_ssl). These settings will force HTTP requests to use HTTPS, terminate HTTPS connections with TLS and strong ciphers, and connect to Rundeck at 127.0.0.1 instead of the public interface.

/etc/apache2/sites-available/rundeck-redirect-port80.conf

```
<VirtualHost _default_:80>
ServerName rundeck-prod.galactica.test:80
Redirect permanent / https://rundeck-prod.galactica.test/
</VirtualHost>
```


© 2016 The SANS Institute Author retains full rights.

/etc/apache2/sites-available/rundeck-ssl.conf

```
<VirtualHost _default_:443>

ServerName rundeck-prod.galactica.test:443
ServerAlias *.galactica.test

SSLProxyEngine On
SSLEngine On
ProxyPreserveHost On
SetEnv proxy-sendchunked
ProxyVia On
ProxyRequests Off

ErrorLog ${APACHE_LOG_DIR}/ssl_error_log
TransferLog ${APACHE_LOG_DIR}/ssl_transfer_log

LogLevel warn

SSLProtocol TLSv1

SSLCipherSuite HIGH:!aNULL:!MD5
SSLHonorCipherOrder On

SSLCertificateFile /etc/ssl/certs/rundeck-apache.crt
SSLCertificateKeyFile /etc/ssl/private/rundeck-apache.key

RequestHeader set Front-End-Https "On"
RequestHeader set X-Forwarded-Proto "https"
RequestHeader set X-Forwarded-Port 443
Header add Strict-Transport-Security "max-age=631138519; includeSubdomains;
preload"
Header add X-Frame-Options SAMEORIGIN
Header add X-Content-Type-Options nosniff
Header add X-XSS-Protection "1; mode=block"

ProxyPass / http://127.0.0.1:4440/  keepalive=On
ProxyPassReverse / http://127.0.0.1:4440/

</VirtualHost>
```


© 2016 The SANS Institute Author retains full rights.

With the configuration files in /etc/apache2/sites-available, remove any defaults from /etc/apache2/sites-enabled and copy over the new rundeck configs. Restart Apache to load new configs.

```
cd /etc/apache2/sites-enabled
sudo rm 000-default.conf
sudo ln -s /etc/apache2/sites-available/rundeck-* .
```

Configure the "grails.serverURL" value in /etc/rundeck/rundeck-config.properties to accept requests for the HTTPS URL. Restart Rundeck to load the config.

```
# change hostname here
grails.serverURL=https://rundeck-prod.galactica.test
```

Finally, set the "-Drundeck.jetty.connector.forwarded=true" in /etc/rundeck/profile to retain the XFF header information (see http://rundeck.org/docs/administration/configuring-ssl.html#using-an-ssl-terminated-proxy for more information). Rundeck now has TLS, strong-cipher termination in Apache. Additional modules are available for multi-factor authentication (e.g. Google Authenticator) and web application firewall protection (e.g. ModSecurity).

2.4. Node Definitions

Nodes are servers, VMs, instances, or devices that Rundeck can access via SSH or connection plugins. The default option for node definitions is to use the built-in provider that is managed by XML files. Small and relatively static environments function fine with the default provider. Larger, dynamic networks benefit from node provider plugins such as the AWS EC2 or PuppetDB plugins.

One of the more compelling features of Rundeck is how it handles metadata. Rundeck nodes have attributes that describe the instance (e.g. Name, IP address, operating system, etc.). The node provider manages attributes in a key value format (e.g. the EC2 plugin will populate the AWS EC2 attribute "instanceId=i-389f2cdk" for a corresponding Rundeck node). Tags are a type of attribute used for classifications or categories (Rundeck User Guide, 2016). The combination of tags and attributes allow for dynamic targeting of hosts for jobs and ad-hoc commands.

John Becker, jbecker42@gmail.com

© 2016 The SANS Institute Author retains full rights.

2.5. Connectivity

A wide range of plugins is available for Rundeck to expand access beyond the default SSH connectivity (Rundeck.org, n.d.). The WinRM (Windows Remote Management) plugin is available for native Windows commands. An alternative is to use SSH on Windows with OpenSSH server or similar. In other cases, Rundeck functions as a front-end for Puppet, Ansible, or Chef commands. Careful review is needed to determine what connectivity and access Rundeck should have in an organization. On one end of the spectrum, Rundeck could have root or admin-equivalent privileges on all devices and rely exclusively on internal ACLs for controls. Other teams could use multiple Rundeck instances with restricted service accounts to reduce the impact of a Rundeck server compromise.

2.6. Key Management

Rundeck's Key Storage system stores private keys as either local files or as BLOBS in the attached database. Neither solution is encrypted unless using a Storage Converter plugin (Rundeck Administrator Guide, 2016). Storing keys in an external database such as MySQL makes them available for multiple Rundeck instances but also has an increased attack surface. Use the Storage Converter plugin for encrypting keys kept in the database.

Using the filesystem storage for the Rundeck Key Storage with the Storage Converter for encryption is possible, but the tradeoff between security and availability may not be worth it. Restarts of Rundeck would require reading the Storage Converter password from a local source (file, dongle, etc.) or manually typed in at startup. Either way, service startups are more complicated, and the encryption key is potentially accessible in memory or on disk. A reasonable balance with confidentiality and availability is to encrypt the local filesystem within the OS or Virtual Machine and keep the Rundeck Key Storage in cleartext.

© 2016 The SANS Institute Author retains full rights.

2.7. Jobs and Commands

Rundeck provides a user interface and API for executing and scheduling commands or jobs. If something works over SSH, then it is most likely going to work as a Rundeck job or command. Ad-hoc commands are singular executions shell commands against target nodes. They can be as simple as 'uptime' or a complex string of piped commands. Jobs are an ordered set of steps comprised of CLI commands, shell scripts, and other Rundeck jobs. When sequenced into a job, these steps can perform complex orchestrations such as load-balancer failovers, operating system patching, and test script executions.

Jobs can take variable input with "options" that are entered via the UI or from external option providers (e.g. Jenkins). Similarly, a job step script can be a remote file accessed via a file path or URL. Remote options and scripts improve flexibility with version control, but also introduce added complexity for securing the content. Instead of just securing Rundeck, remote option and script providers are also in scope for hardening and audit.

2.7.1. Scheduling

Many Rundeck users focus on the scheduling feature (Edwards, 2014). Scheduled jobs execute at set times similar to cron services in Linux or schtasks in Windows. Using Rundeck for scheduling centralizes visibility into the tasks and outputs running on a network. Incident responders will look for scheduled tasks on hosts as possible logic bombs or other signs of intrusion (Kral, 2011). Moving scheduled tasks to Rundeck allows disabling cron and schtasks services. In turn, this makes detecting unauthorized scheduled tasks easier on hosts.

2.7.2. Job Naming

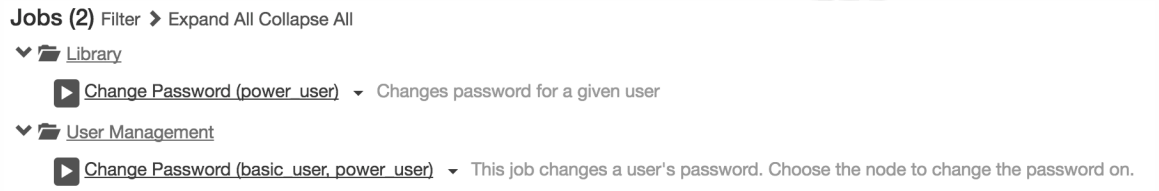
Naming jobs may seem straightforward at first until an extensive collection of jobs and roles exist in Rundeck. As the jobs and ACL policy files grow, the Rundeck administrator becomes a bottleneck to align access controls with jobs. This bottleneck leads to contention for group names and rigid rules for job creation. An alternative method is to include the role names within the job name itself. A job called "Change Password" would become "Change Password (basic_user, power_user"). The keywords

© 2016 The SANS Institute Author retains full rights.

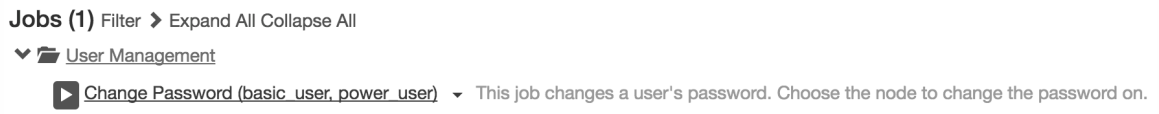
basic_user and power_user are in regex patterns inside ACLs. Anytime a job is added or changed it need only include the correct role keyword to be usable. This layout significantly reduces the overhead for managing ACL policies.

Below are screenshots of the same set of jobs but with ACLs limiting “read” access for the basic_user role. In this situation, the power_user role can execute both jobs directly, but the basic_user role can only view “Change Password (basic_user,power_user).”

power_user View



basic_user View



2.7.3. Job Groups

A good starting point for grouping jobs is to separate out jobs by function. User Management, Release, Patching are all examples of good top-level groups. A "Library" group of jobs can be referenced by multiple other jobs while keeping the contents of the scripts hidden. For example, a generic "Change Password" job can be created as a parent job that is executed by users. This job will target a node and pass a "username" option to a Library job that is typically not viewed directly by users. The Library job contains the actual scripts for executing the commands. Basic user ACLs reference the Library group for "run" but not "read" access. This approach enables execution while preventing most users from viewing any sensitive information.

© 2016 The SANS Institute Author retains full rights.

2.8. Access Control Lists

Access Control Lists (ACLs) in Rundeck permit role-based access with a high degree of granularity. Resources in Rundeck are denied by default until explicitly allowed by ACLs. ACL policies manage privileges at many levels within Rundeck including projects, jobs, nodes, ad-hoc commands, key storage, and the Rundeck API itself. Policy files are written in YAML and reside in the /etc/rundeck directory, but can also be created using the System ACLs API and the Project ACLs API.

For a straight-forward and easily audited set of ACLs, start with standard .aclpolicy files in the /etc/rundeck directory. Create one .aclpolicy file per group or role and name the file to match (e.g. an ACL policy file for the group "operations" should be called "operations.aclpolicy"). Test the resulting ACLs after any changes.

Roles for example ACLs

Role	ACL File Name	Project Access	Node Access	Job Access
Power User	power_user.aclpolicy	All	All	Create, update, run
Admin	admin.aclpolicy	All	All	All
Basic User	basic_user.aclpolicy	Support, Network	Production	"basic" jobs

© 2016 The SANS Institute Author retains full rights.

2.8.1. ACL Breakdown

This is the default admin.aclpolicy from /etc/rundeck enables full access to everything for any member of the "admin" group.

```
description: Admin, all access.
context:
 project: '.*' # all projects
for:
 resource:
  - allow: '*' # allow read/create all kinds
 adhoc:
  - allow: '*' # allow read/running/killing adhoc jobs
 job:
  - allow: '*' # allow read/write/delete/run/kill of all jobs
 node:
  - allow: '*' # allow read/run for all nodes
by:
 group: admin

---

description: Admin, all access.
context:
 application: 'rundeck'
for:
 resource:
  - allow: '*' # allow create of projects
 project:
  - allow: '*' # allow view/admin of all projects
 project_acl:
  - allow: '*' # allow admin of all project-level ACL policies
 storage:
  - allow: '*' # allow read/create/update/delete for all /keys/* storage content
by:
 group: admin
```

Rundeck ACL policy files define what actions (read, create, update, delete, admin, enable_executions, disable_executions, configure, import, export) apply to resources (project, system, system_acl, user, job, storage, job, node, ad-hoc, or event). The full

© 2016 The SANS Institute Author retains full rights.

options for resources and actions are available at http://rundeck.org/docs/administration/access-control-policy.html. Options available within ACLs make for accurate, if not complicated, rulesets. Structuring the ACL policy files for flexibility can prevent frustration in job management. The approach recommended in this paper is as follows:

Deny Rundeck server access. By default, this is "localhost" and can be referenced in the ACL with the "nodename" node resource property.

```
node:
  - match:
      nodename: 'localhost'
    deny: '*'
  - match:
      nodename: '.*'
    allow: '*'
```

Limit Rundeck administration functions to a single role (Admin in the example below). Admin access includes the ability to create/modify projects, modify project ACLs, and modify key storage. The "application: 'rundeck'" code blocks are more restrictive for non-Admin roles.

Admin Access

```
description: Admin, all access.
context:
  application: 'rundeck'
for:
  resource:
    - allow: '*' # allow create of projects
  project:
    - allow: '*' # allow view/admin of all projects
  project_acl:
    - allow: '*' # allow admin of all project-level ACL policies
  storage:
    - allow: '*' # allow read/create/update/delete for all /keys/* storage content
by:
  group: admin
```

John Becker, jbecker42@gmail.com

© 2016 The SANS Institute Author retains full rights.

Basic User Access

```
description: Basic User, restricted access.
context:
 application: 'rundeck'
for:
 resource:
  - equals:
      kind: system
    allow: [read] # allow read of resources
 project:
  - match:
      name: ['Support','Network']
    allow: [read] # Allow read access to specific projects
 storage:
  - allow: [read] # Allow allow read access for using ssh keys
by:
 group: basic_user
```

Use regular expressions to target specific access to keywords in Job names. Deny read access to prevent users from viewing jobs and projects they should not access. Their user interface will be cleaner, and there are fewer chances for accidental privilege escalation. The example below will enable the basic_user role to run any job with the keyword “basic_user” in the name as well as prevent viewing of jobs in the Library group.

© 2016 The SANS Institute Author retains full rights.

Basic_user ACL

```
description: Basic User, restricted access.
context:
 project: '.*' # all projects
for:
 resource:
  - allow: '*' # allow read/create all kinds
 adhoc:
  - allow: '*' # allow read/running/killing adhoc jobs
 job:
  - match:
     name: '.*basic_user.*'
    allow: [read,run,kill] # allow read/run/kill of jobs
  - match:
     group: 'Library.*'
    allow: [run,kill] # allow read/run/kill of all jobs
 node:
  - allow: '*' # allow read/run for all nodes
by:
 group: 'basic_user'
```

2.9. Logging and Execution History

Rundeck has excellent history and full output logging features (Oyster.com Tech Blog, 2016). Standard output from commands is available as "activity" along with the time, date, and user who executed the job or ad-hoc command. Activity history resides in the database and is searchable via the Rundeck user interface.

Rundeck uses log.4j and logs write to /var/log/rundeck in the package distribution. Key logs for security review are rundeck.audit.log (ACL decisions), rundeck.jobs.log (changes to jobs), rundeck.log (general application messages included execution activity). Full details on Rundeck logging and formatting are available at http://rundeck.org/docs/administration/logging.html.

3. Example Use Case for Incident Response

While Rundeck supports many different use cases, incident response is a good example for security teams. Incident handling has 6 phases: Preparation, Identification,

© 2016 The SANS Institute Author retains full rights.

Containment, Eradication, Recovery, and Lessons Learned (Kral, 2011). Rundeck fits within the Preparation (Access Control and Tools), Identification (Event gathering), and Containment (Short-term Isolation, Backup, Long-term Isolation) phases. Rundeck could also prove useful in the Eradication and Recovery phases as a Build/Deployment/Release tool. For this example, Rundeck is used for accessing, finding, containing, and removing intruders.

3.1.1. Preparation

The Preparation phase focuses on team readiness for handling an incident with little or no notice (Kral, 2011). Rundeck is a useful tool to access and review systems remotely. Incident response teams can build regular jobs to retrieve logs, hash critical files, check permissions, as well as make changes such as update firewall rules or apply patches. Many scripts used for hunting and investigations work well as Rundeck jobs.

3.1.2. Identification

A responder can use the Rundeck jobs created in the Preparation phase to determine whether an incident has occurred. These jobs are helpful for any systems or patterns not included in IDS systems. Ad-hoc commands are useful for hunting for specific patterns across many devices. For example, the ad-hoc command below will return the hostname, md5sum for /etc/init/ssh.conf, and the number of files in /etc/init in a CSV format. Copy and save the output as a CSV file for analysis.



3.1.3. Containment

The Containment phase goal is to prevent further damage and to preserve evidence (Kral, 2011). Compromised nodes tagged with a unique identifier, such as the Incident ID "inc50298" in this example, are easy to group in Rundeck. Containment jobs can then be

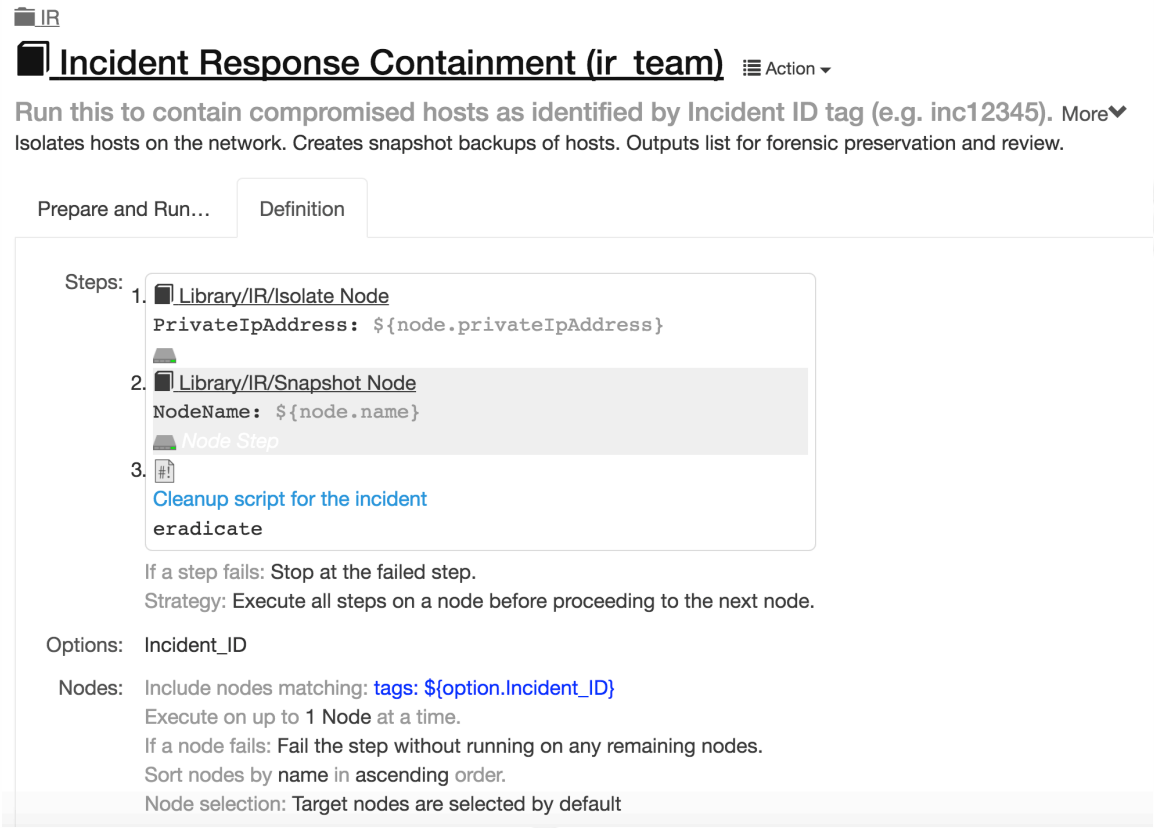
© 2016 The SANS Institute Author retains full rights.

targeted against nodes by Incident ID when “tags: ${option.Incident_ID}” is set for the node target. Short-term containment can be as simple as updating firewalls or network devices to stop traffic to a given host. Preservation can take the form of a snapshot or backup of a compromised host. Finally, long-term containment step is to remove any backdoors or malware to return the device to production use. Below is an example Rundeck containment job:

1. Step 1 is a Library job reference. The target node's IP address passes to the Library job "Library/IR/Isolate Node". The Isolate Node job will update the firewall rules to block traffic to the IP address.
2. Step 2 performs a snapshot of the node from the backup server. The Library job “Library/IR/Snapshot Node” runs a snapshot script for the given hostname.
3. Step 3 uses a script path "file:///ir/cleanup/${option.Incident_ID}.sh" to reference a unique script for this incident cleanup. The path will expand to the filesystem location on Rundeck as “/ir/cleanup/inc50298.sh". In this script are containment commands unique for this incident. Running the job with an empty script is fine if full cleanup steps are not ready.

©2016 SANS Institute, Author retains full rights.

© 2016 The SANS Institute Author retains full rights.



4. Conclusion

Rundeck provides a user-friendly interface for scheduling and executing commands against a dynamic inventory of devices. Central, agentless access to network devices, Windows servers, Linux VMs, and cloud instances is very powerful for developers, admins, and security teams alike. All teams benefit from increased visibility to execution history, logging, and centralized task scheduling. The coordination between security teams and DevOps teams have improved collaboration with access control and key management.

However, a single point of access for systems is also a key target for intruders. Protecting this access begins with hardening Rundeck itself using TLS and permission changes. A common approach to job and ACL layout ensures correct access, but is still agile, for modern environments. Maintaining clear roles and ACL mappings results in low-stress least privilege access.

John Becker, jbecker42@gmail.com

© 2016 The SANS Institute Author retains full rights.

As teams adopt Rundeck for everyday tasks, the history of those tasks becomes more valuable for auditors and incident responders. Furthermore, executions outside of Rundeck are highlighted for possible investigation. Rundeck itself is useful during incident response for gathering information and containment. Security teams can quickly tag compromised nodes and run targeted discovery and containment jobs. This approach works with other tools and scripts. However, the built-in features in Rundeck make for a simple solution without requiring extensive customization.

©2016 SANS Institute, Author retains full rights.

© 2016 The SANS Institute Author retains full rights.

References

Oyster.com Tech Blog. (2016, March 7). *Rundeck vs. Crontab: Why Rundeck won*. Retrieved July 29, 2016, from http://tech.oyster.com/rundeck-vs-crontab-why-rundeck-won/

Class UnixCrypt. (2013, January 7). Retrieved July 28, 2016, from https://commons.apache.org/proper/commons-codec/apidocs/org/apache/commons/codec/digest/UnixCrypt.html

Critical Security Controls for Effective Cyber Defense. (2015, October 15). The Center for Internet Security, 6.0. Retrieved March 12, 2016, from https://www.cisecurity.org/critical-controls/

Dobbertin, H. (1996). *The Status of MD5 After a Recent Attack. CryptoBytes,* 2(2), 1-6. Retrieved from ftp://ftp.arnes.si/packages/crypto-tools/rsa.com/cryptobytes/crypto2n2.pdf.gz.

Edwards, D. (2014, August 29). *Growing popularity of Rundeck's job scheduler features*. Retrieved July 21, 2016, from http://rundeck.org/news/2014/08/29/Rundeck-job-scheduler.html

Hu, V. C., Ferraiolo, D. F., & Kuhn, D. R. (2006, September). *Assessment of Access Control Systems*. Retrieved July 29, 2016, from http://csrc.nist.gov/publications/nistir/7316/NISTIR-7316.pdf

Jetty, *Secure Password Obfuscation*. (2016, July 27). Retrieved July 28, 2016, from https://www.eclipse.org/jetty/documentation/9.3.x/configuring-security-secure-passwords.html

Jetty Source Code, *Password.java.* (2016, February 9). Retrieved July 28, 2016, from https://github.com/eclipse/jetty.project/blob/c99c02e2f59cc4c65cc9b893710e48eeeb3bef0b/jetty-util/src/main/java/org/eclipse/jetty/util/security/Password.java

Kral, P. (2011, December 5). *The Incident Handlers Handbook*. Retrieved July 14, 2016, from https://www.sans.org/reading-room/whitepapers/incident/incident-handlers-handbook-33901

Lawler, E. (2015, April). *Is DevOps Breaking Your Company?* Retrieved July 15, 2016, from https://www.rsaconference.com/writable/presentations/file_upload/asd-w02-is-devops-breaking-your-company.pdf

John Becker, jbecker42@gmail.com

© 2016 The SANS Institute Author retains full rights.

Rome, J. (n.d.). *Enclaves and Collaborative Domains.* Retrieved July 29, 2016, from http://web.ornl.gov/~webworks/cppr/y2001/pres/117259.pdf

Rundeck Administrator Guide. (2016, June 10). Retrieved July 28, 2016, from http://rundeck.org/docs/administration/index.html

Rundeck User Guide. (2016, August 3). Retrieved August 8, 2016, from http://rundeck.org/docs/manual/index.html

Rundeck.org. (n.d.). Retrieved July 11, 2016, from http://rundeck.org/

Schneier, B. (1999, November 19). *A Plea for Simplicity*. Retrieved July 28, 2016, from https://www.schneier.com/essays/archives/1999/11/a_plea_for_simplicit.html

Schueler, G. (2015, April 16). *Preferred database backend for Rundeck*. Retrieved July 31, 2016, from https://groups.google.com/forum/#!topic/rundeck-discuss/zK4CYRYGTVA

Swanson, M., & Guttman, B. (1996, September). *Generally Accepted Principles and Practices for Securing Information Technology Systems*. Retrieved July 12, 2016, from http://csrc.nist.gov/publications/nistpubs/800-14/800-14.pdf

© 2016 The SANS Institute Author retains full rights.