Description: datacom-logo

Sudoers Configuration Propagation

Initial Proposal



Ben Schofield, Unix Systems Engineer

Datacom Systems Limited

18 July 2014

Version 0.1

CONFIDENTIAL

Document Information

Version History

The following table shows the version history for this document.

| Version | Date | Author(s) | Description of Change |
| --- | --- | --- | --- |
| 0.1 | 18/07/2014 | Ben Schofield | Initial Version |
|  |  |  |  |
|  |  |  |  |

Contact Person Details

Any questions regarding this document should be directed to:

Name Ben Schofield  
 Unix Systems Engineer  
Email: ben.schofield@datacom.co.nz  
Phone: +64 (0) 27 702 8034

Commercial in Confidence

This document contains intellectual property and proprietary information belonging to Datacom Systems Limited. The content of this document must not be copied, re-used or disseminated in whole or in part without the express prior permission of Datacom Systems Limited. All information is confidential and provided in commercial confidence.

Reviewers

The following table shows the version history for this document.

| Name | Comment/Finding |
| --- | --- |
| Domingos Novo | Did not find any issues. |

Signoff

| Name | Role | Signature | Date |
| --- | --- | --- | --- |
|  |  |  |  |

Distribution list

| Name | Role | Date |
| --- | --- | --- |
|  |  |  |

Table of Contents

[1 Introduction 4](#_Toc393783678)

[1.1 Document Purpose 4](#_Toc393783679)

[1.2 Scope 4](#_Toc393783680)

[1.3 Document References 4](#_Toc393783681)

[1.4 Definitions 4](#_Toc393783682)

[2 Current Method 5](#_Toc393783683)

[2.1 Process Description 5](#_Toc393783684)

[2.1 Risks and Inefficiencies 5](#_Toc393783685)

[3 Proposed Solution 6](#_Toc393783686)

[3.1 Process Description 6](#_Toc393783687)

[3.2 Risks and Mitigations 7](#_Toc393783688)

[3.2.1 Compromised Intermediate Server 7](#_Toc393783689)

[3.2.2 Master sudoers File Corruption 7](#_Toc393783690)

[3.2.3 sudoers File Corruption During Transfer 8](#_Toc393783691)

[4 Further Enhancements 9](#_Toc393783692)

[4.1 Semi-autonomous Editing 9](#_Toc393783693)

# Introduction

## Document Purpose

The purpose of this document is to describe the proposed changes to the NZDF Linux sudoers file administration environment. Inefficiencies have been identified in the configuration and distribution of the sudoers files, and security could be improved through the increased oversight.

## Scope

This document is intended to cover a high level design of sudoers distribution and management. It does not cover implementation or operational details.

## Document References

| Document | Version | Relevance |
| --- | --- | --- |
|  |  |  |
|  |  |  |

## Definitions

| Term | Acronym | Definition |
| --- | --- | --- |
| sudoers |  | The sudoers file (/etc/sudoers) grants logged in users the right to privilege escalation for specific commands. |
| sudo |  | The Linux application that reads the sudoers file. |
| New Zealand Defence Force | NZDF |  |
| Defence Force Order | DFO | A series of NZDF documents detailing required security and operational requirements. |
| Secure Shell | SSH | A method of securely accessing a remote system via an encrypted connection. |
| Secure File Transfer Protocol | SFTP | A method of transferring files securely through an SSH tunnel. |
| Access Control List | ACL | An extension of Linux file permissions which gives more granular control of file access. |

# Current Method

## Process Description

Currently, each sudoers file is managed manually on a per server, per user, per command basis. There are opportunities for legacy users and inconsistent sudoers configurations across servers. Each server must be logged into individually, privileges elevated, sudoers edited and saved whenever a sudoers change is required. Figure 1 illustrates a high level view of this process with three servers. The connections are commonly made over SSH, but console access is also possible, so the connection process is unlabelled here.

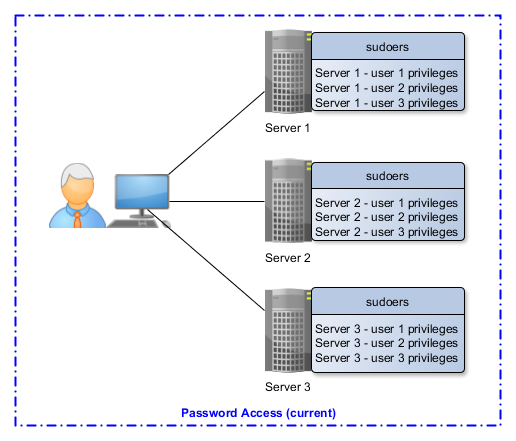


Figure 1 - Current sudoers management method

## Risks and Inefficiencies

When a new user joins, the process for adding them to each relevant sudoers file is time consuming, repetitive and open to error. Because the process is time consuming, it’s common to grant a user more privileges than might otherwise be required, such as allowing the user to switch users entirely when their only requirement might be to, for instance, restart a service.

As an unfortunate side effect of switching users, auditing capabilities are reduced. Auditing the changes and recording the time of those changes is still accurate, however the changes are attributed to the switched-to user. It is common for many users to be able to switch to an alternate single user, such as the user that runs the database service. Therefore, it would be very difficult to accurately establish which real users made system modification.

# Proposed Solution

## Process Description

In the proposed system, an Intermediate Server is introduced. The Intermediate Server could be an existing server or a new server. Its only purpose is to act as a repository for the master sudoers file. The master sudoers file is the only sudoers file that is edited by an administrator. The master file is then distributed to other servers, so that each server has an identical sudoers file.

The method of sudoers distribution proposed is to use public key authentication over SSH. Public key authentication is generally more secure than using a password because, unlike a password, a key cannot be guessed. It does not matter which side initiates the connection - the Intermediate Server could hold keys for accessing the other servers and push the sudoers configuration file to them, or the other servers could hold a key to the Intermediate Server and pull the sudoers configuration from it. The end result is the same, but the execution and methods of security are slightly different.

Figure 2 gives an example of the proposed sudoers distribution system.

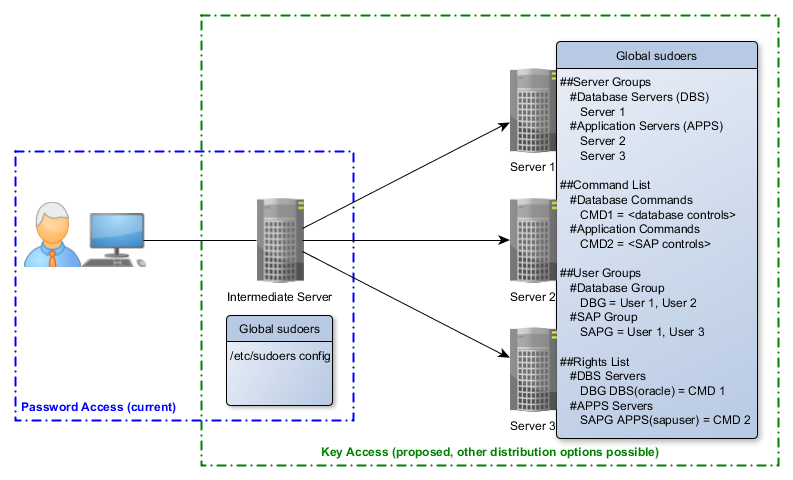


Figure 2 - Proposed sudoers management method

Also illustrated in Figure 2 is a logical representation of the new sudoers file, which briefly describes how a single sudoers file would operate across multiple servers. Servers are grouped into ‘roles’, such as Database servers (DBS) and Application servers (APPS). A single server can also have multiple roles (such as being an Application server and a Database server).

Commands are firstly defined with aliases. This is for tidiness and clarity in the Rights List. For example, ‘/etc/init.d/oracle restart’ could become OracleRestart, although Figure 2 shows more generic examples, CMD1 and CMD2, due to space limitations.

Like servers, users are grouped into their main function, such as a Database Group (DBG) or a SAP Group (SAPG). User functions can also overlap and one user can be part of multiple groups to inherit the privileges of each. For example, in Figure 2, User 1 has inherited privileges of both the Database Group and the SAP Group.

Finally, the users’ rights are defined in the Rights List. This takes the basic format of all sudoers files where a user/group is firstly defined, followed by the server/group of servers, with the user that the command it to be executed as in brackets, and finally followed by the command. In the first example, all Database Group users can execute the CMD1 aliased command on all servers in the Database Servers Group as the user ‘oracle’.

With the sudoers file built this way, adding and removing users, changing user IDs, changing the user the commands are run as, adding new servers and adding or modifying commands for the user to run is a trivial, 30 second process. Auditing of the sudoers file across all servers is also significantly improved, as is the auditing of the users that ran commands.

## Risks and Mitigations

### Compromised Intermediate Server

#### Risk

There is a risk that, should the Intermediate Server be compromised by an unauthorised entity, the sudoers file could be maliciously modified and allow elevated rights for users on the remote servers.

#### Mitigation

By default, the Intermediate Server is no more susceptible to compromised than the existing servers. Should a malicious user have the will, means and opportunity to modify the sudoers file on the Intermediate Server, the attack process would be no different than for an existing server that’s already in use. Furthermore, if the chosen Intermediate Server is not an existing machine that’s used by general users, the server could be secured well beyond the levels of a ‘normal’ server.

Possible security improvements that could be extended to a ‘high security’ system that a ‘normal’ system could not (due to operational restrictions) are, for example, access only from specific hosts, access only with specific keys, access at only specific times, and access only to specific users (as opposed to all users in Active Directory). Built this way, using an Intermediate Server could be more secure than the current method of sudoers modification, especially when local ‘root’ administration is forced through the use of sudo.

### Master Sudoers File Corruption

#### Risk

There is a risk that the master sudoers file may become corrupt or otherwise lost, or deleted, which would prevent distribution or worse, distribute a broken sudoers file. A broken sudoers file would prevent users from elevating their privileges.

#### Mitigation

There is no increased risk of sudoers corruption than any other file on the file system. Appropriate backups and version control systems are effective at mitigating problems of corruption. Local copies (on the Intermediate Server) of the sudoers file stored on different disks/LUNs may also speed up the restore process of a broken sudoers file.

### Sudoers File Corruption During Transfer

#### Risk

The sudoers file may corrupt during transfer from the Intermediate Server to the remote server. A broken sudoers file would prevent users from elevating their privileges.

#### Mitigation

As with local file corruption, sudoers is no more susceptible to corruption during transit than any other file. To mitigate this risk, a checksum of the sudoers file can be completed at the sending and receiving end. Some file delivery protocols support this as an optional flag, such as rsync.

# Further Enhancements

## Semi-autonomous Editing

To further increase security, reliability and auditability of sudoers edits, it is possible to offload the modification of the sudoers file to an autonomous build process which collects necessary values from a database. This removes the need for direct sudoers access for administrators, means that the sudoers file is always syntactically correct and means that every sudoers entry is auditable, as opposed to just sudoers file edits like in the current and proposed systems. Using sudoers this way could mean that root access may never be required, as any change requiring root level privileges could be temporarily granted to a user

By introducing a system through which edits are submitted, administrators can sign off each other’s submissions. For example, Administrator 2 adds a User to a group. Before the change is merged into the distributed sudoers file, approval from Administrator 1 must be sought, and Administrator 1 would sign off the change as both sufficient and appropriate for the rights required by the new User. Figure 3 illustrates a high level design for this system.

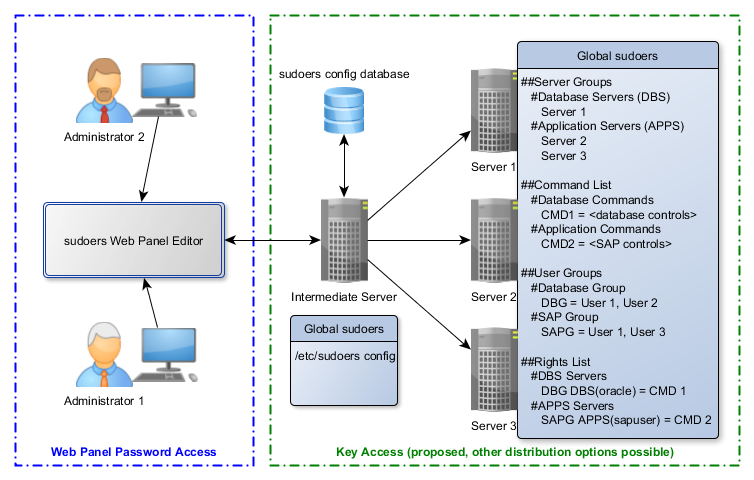


Figure 3 - Potential further enhancements for sudoers management

The introduction of a sudoers modification and distribution infrastructure like the one illustrated in Figure 3 could also be expanded to have single or grouped servers execute commands collectively, such as mounting a new NFS share across many systems in unison.