

Automating Security Policy Implementation

Jarrold Millman
UC Berkeley
Neuroscience Institute

Automating Security Policy Implementation

1

4th Annual Security IT Conference
Anaheim, California

Thursday, March 23, 2006
Concurrent Session IV 9:00-10:00 a.m.

Jarrold Millman
Helen Wills Neuroscience Institute
132 Barker Hall, MC #3190
Berkeley, CA 94720-3190

[http://cirl.berkeley.edu/
jarrold.millman@gmail.com](http://cirl.berkeley.edu/jarrold.millman@gmail.com)

Overview

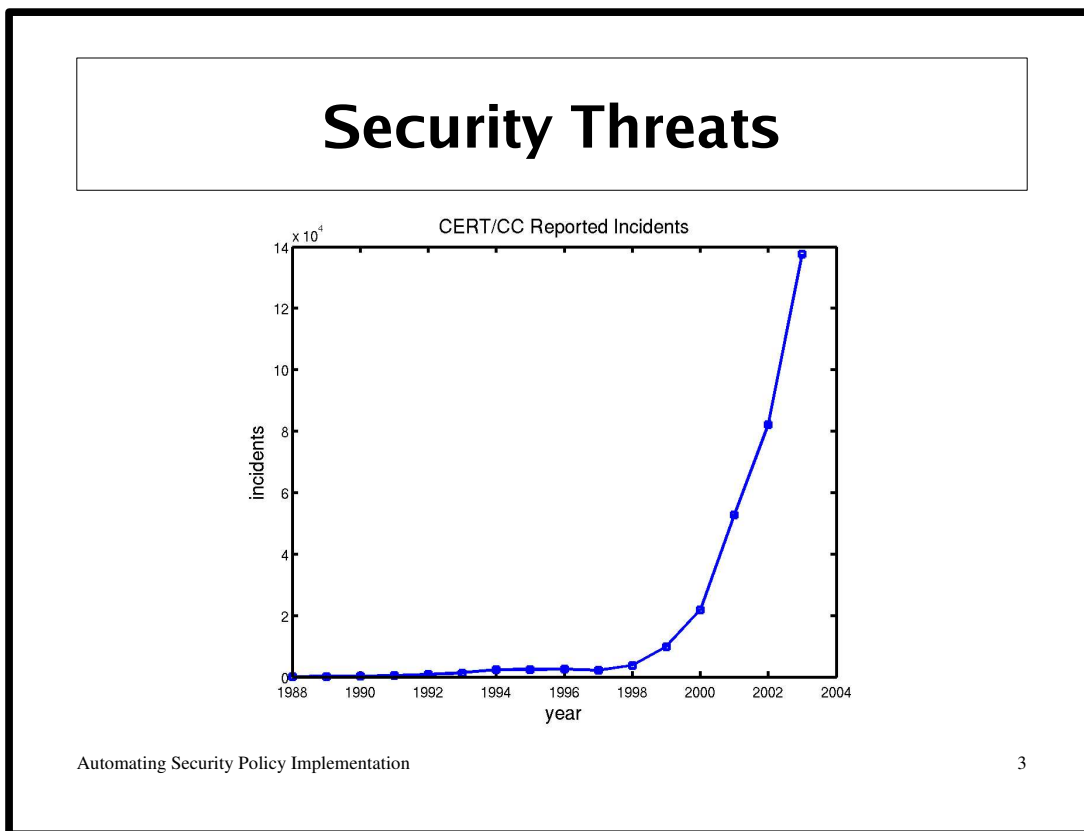
- Security Policy
 - Automation & Configuration Management
 - Architecture, Deployment, Maintenance
- UCB Brain Imaging Center
 - Fedora Linux
 - Kickstart & cfengine

An organizations security policies form the basis for protecting its information resources. However, to be of practical value in preventing security compromises, these policies must be implemented and reliably deployed. In this presentation, I will begin with a general discussion about security policy and then focus on how we automate security policy implementation on Linux hosts at UC Berkeley's Brain Imaging Center (BIC).

Security discussions often focus on how a host should be configured or how to configure certain security applications, but say little about how to ensure that the system will maintain that configuration. Yet, over time a system will change due to ordinary use, system maintenance, regular updates, as well as a variety of other causes.

First, I will introduce the basic ideas related to automated deployment and configuration, starting with a brief discussion of security policy and the ideas surrounding automated implementation, including client-server architectures and pull versus push systems. Then, I will give a detailed account of how Berkeley's BIC automates its systems deployment and management with cfengine.

Though I will focus mostly on our experience with Linux systems, cfengine runs on most UNIX-like hosts (e.g., Solaris, Linux, MacOS X) as well as Windows 2000 hosts and above. I will cover (1) the components of cfengine and how they are used, (2) its security and trust model, (3) how to install & configure it, and (5) how to customize it for security tasks. Using cfengine and other tools, I will show how to report on suspicious files and directories, maintain filesystem permissions, monitor file checksums, disable network services, enforce password policy, patch updates, as well as more advanced topics.



CERN Incidents Reported

“Given the widespread use of automated attack tools, attacks against Internet-connected systems have become so commonplace that counts of the number of incidents reported provide little information with regard to assessing the scope and impact of attacks. Therefore, as of 2004, we will no longer publish the number of incidents reported. Instead, we will be working with others in the community to develop and report on more meaningful metrics, such as the 2004 E-Crime Watch Survey. We welcome ideas and proposals for other collaborations in this area.”

http://www.cert.org/stats/cert_stats.html

Common Tactics

- Denial of Service (DoS)
- Buffer Overflows
- Trojan Horses
- Physical Access
- Intercepted Communications
- Social Engineering
- Lack of User Cooperation

Security

- Security is a process
- Security is about trade-offs

Automating Security Policy Implementation

4

"The superior man, when resting in safety, does not forget that danger may come. When in a state of security he does not forget the possibility of ruin. When all is orderly, he does not forget that disorder may come. Thus his person is not endangered, and his States and all their clans are preserved."

Confucius (551 BC - 479 BC)

General Security

<http://www.cve.mitre.org/>

<http://www.cert.org/>

<http://www.sans.org/>

<http://www.securityfocus.com/>

Fedora/Redhat Security

<http://www.redhat.com/security/>

Security Policy

- Authentication
- Authorization
- Data Protection
- Application Configuration

Automating Security Policy Implementation

5

The SANS Security Policy Project

<http://www.sans.org/resources/policies/>

From the "SAGE Short Topics Booklets Series"

#2 A Guide to Developing Computing Policy Documents

Edited by Barbara L. Dijker

#3 System Security: A Management Perspective

by David L. Oppenheimer, David A. Wagner, and Michele D. Crabb

Edited by Dan Geer

See the following ISO standard

ISO17799

ISO27001

Trust

- Who do you trust?
- What do you trust them with?

Security Policy Automation

- Architecture
 - Client vs. Server
 - Push vs. Pull
- Deployment
 - Cloning vs. Scripting
- Configuration
 - Static vs. Dynamic
 - Central vs. Distributed

Convergent Configuration

- Standard, centralized configuration
- Separate hosts into classes
- Ensure that any necessary host changes are recorded
- Eventually make all necessary changes

Automating Security Policy Implementation

8

I will discuss recent work in automated configuration theory, which argues that a configuration system should satisfy a convergence property. That is the system should periodically run on a given host computer and, if needed, modify the hosts configuration so that it more closely resembles the host's policy specification. I will argue that to be of practical use in a large environment an automated system should:

1. Provide a standard, centralized configuration that all hosts can use.
2. Separate hosts into classes based a variety of factors in order to differentiate desirable behavior.
3. Ensure that any necessary host changes are recorded in such a way that they will be performed again, if needed.
4. Eventually make all necessary changes even for systems with intermittent uptime or network connectivity.

Preliminary Setup

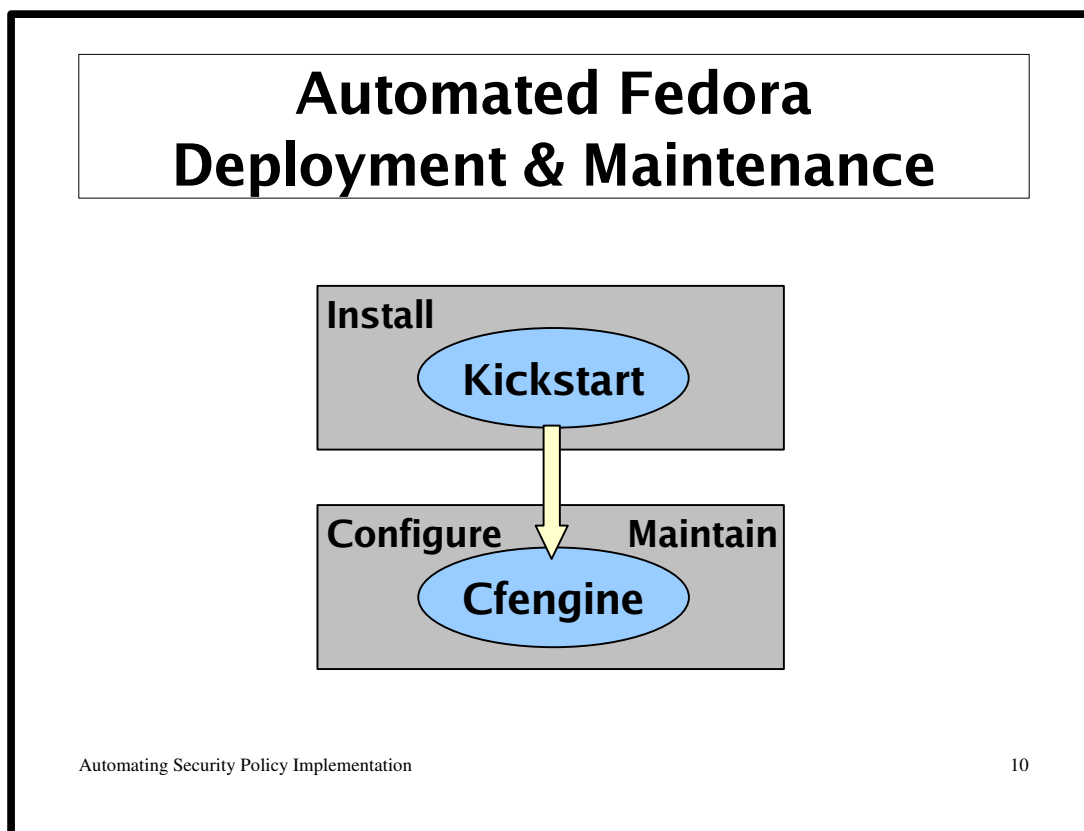
- Physical Security
 - Secure work area
 - Lock-down laptops
- BIOS
 - Set BIOS password
 - Boot order

Automating Security Policy Implementation

9

HAAS Business School

<http://edition.cnn.com/2005/TECH/03/29/stolen.laptop/>



Automating Security Policy Implementation

10

“Fedora Core is a completely free, user friendly, and secure general purpose platform based on Linux. The Fedora Project is an open source project that pioneers leading edge technology and concepts sponsored by Red Hat and supported by the Fedora community.”
<http://fedora.redhat.com/>

Even though this talk focuses on Kickstart and Cfengine, there are several other tools which will be of interest:

- `cron` is a daemon for Linux or UNIX to execute scheduled commands.
- `expect` is an application for automating scripting of interactive programs such as `ssh` written by Don Libes. See <http://expect.nist.gov/>

Kirk Bauer, the author of `autorpm` and `logwatch`, has written an excellent book on automating system administration tasks:
[Automating UNIX and Linux Administration](#)

Kickstart

- Automates Fedora installs
 - Reads from text file rather than prompting the user
 - Extremely flexible & customizable
 - Mass deployment & system recovery

Kickstart

Please see the RHEL 4 System Administrators Guide:

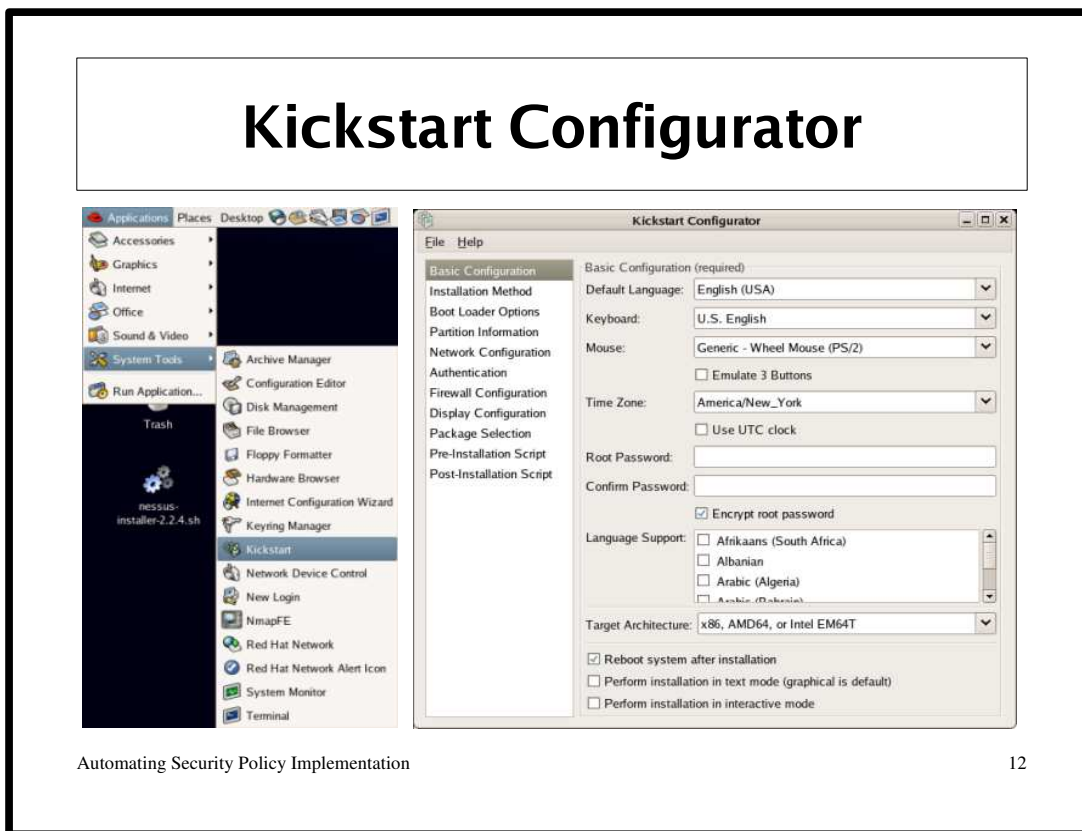
<http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/sysadmin-guide/>

System Installation Suite

SIS is a collection of tools for deploying and configuring Linux installations using mirroring rather than automated installation. See <http://www.systemimager.org/>

Jumpstart

Jumpstart is an automated installation tool for Solaris.



Install

```
sudo yum install system-config-kickstart
```

Run

```
/usr/sbin/system-config-kickstart
```

Kickstart File

- Created either by
 - Kickstart Configuration application
 - Manual
 - Initial install
 - /root/anaconda-ks.cfg

Automating Security Policy Implementation

13

```
# Kickstart file
```

```
install
url --url http://192.168.0.5/fc/4/iso/original-release
lang en_US.UTF-8
langsupport --default=en_US.UTF-8 en_US.UTF-8 en_US en en_US.UTF-8 en_US en
keyboard us
network --device eth0 --bootproto dhcp
rootpw --iscrypted $1$YTOvw5E9$P15AAmrspBNzT1a0sGb62.
bootloader --location=mbr --md5pass=$1$YTOvw5E9$P15AAmrspBNzT1a0sGb62.
firewall --enabled --port=22:tcp
selinux --enforcing
authconfig --enablesshadow --enablemd5
timezone --utc America/Los_Angeles

%packages
@ editors
@ gnome-desktop
@ base-x
@ graphical-internet

%post

# This is where Cfengine should be installed and configured.
#
```

Kickstart Install

- `ks.cfg` accessed by either:
 - Local
 - Floppy
 - CD
 - Network
 - HTTP
 - NFS

Automating Security Policy Implementation

14

To begin a Kickstart installation you boot off FC4 Install CD1 and enter a command at the boot prompt. For example,

Floppy

```
linux ks=floppy
```

CD

```
linux ks=cdrom:/ks.cfg
```

HTTP

```
linux ks=http://<server>/<path>
```

NFS

```
linux ks=nfs:<server>:/<path>
```

Cfengine

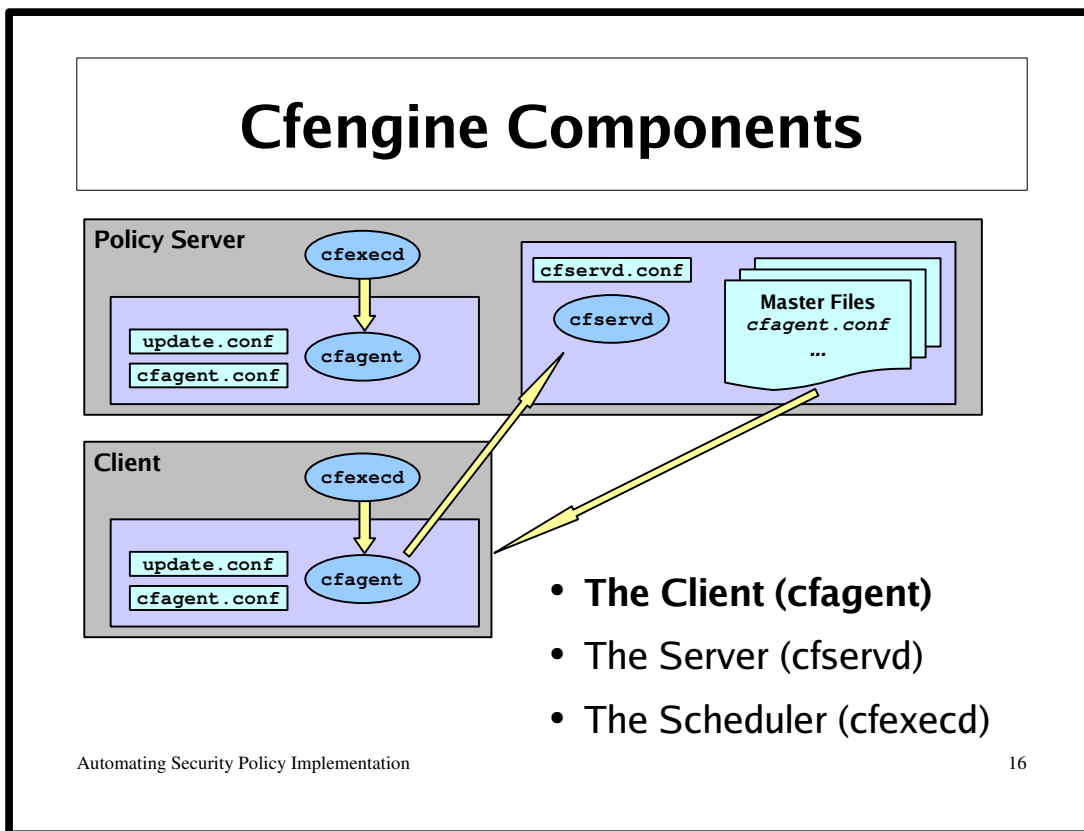
- Cfengine components
- Security & trust model
- Installation & Configuration
- Customizing for security tasks

Cfengine

Cfengine is “an autonomous agent and a middle to high level policy language and agent for building expert systems to administrate and configure large computer networks. Cfengine is designed to be a part of a computer immune system.” The best starting point is the projects homepage: <http://www.cfengine.org/>

Mark Burgess, Professor of Network and System Administration at University College Oslo, Faculty of Engineering, Norway, is the creator and lead developer of Cfengine. Cfengine was released in 1993. In 2002, he released a major rewrite called version 2. He is currently designing version 3, which he plans to release in about 1 year. For more information about Mark's research see: <http://www.iu.hio.no/~mark/>

For more information about automated configuration management see the Large Scale Configuration Management page: <http://homepages.informatics.ed.ac.uk/group/lssconf/>



Terminology

Policy server

Central point of management for cfengine-controlled hosts. Stores all configuration files to be distributed to cfengine clients.

File server

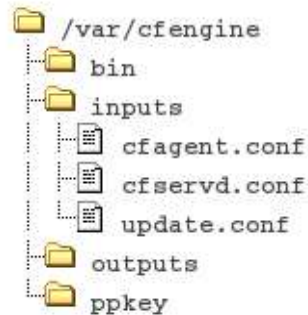
A cfengine host that shares files to cfengine clients.

Policy client/cfengine client

A host controlled by a cfengine policy host.

Cfengine Files

- Default Directory Structure



Automating Security Policy Implementation

17

Components

<code>cfagent</code>	An autonomous configuration agent (<i>required</i>).
<code>cfservd</code>	A file server and remote activation service (<i>optional</i>).
<code>cfexecd</code>	A scheduling and report service (<i>recommended</i>).
<code>cfenvd</code>	An anomaly detection service (<i>strongly recommended</i>).
<code>cfrun</code>	A way of activating cfagent remotely (<i>use this as you need to</i>).
<code>cfshow</code>	A way of examining the contents of helper databases (<i>helper</i>).
<code>cfenvgraph</code>	Ancillary tool for cfenvd (<i>helper</i>).
<code>cfkey</code>	Key generation tool (<i>run once on every host</i>).

Computer Immune System & Anomaly Detection

- The Goal of Cfengine is a Computer Immune System based on
 - Anomaly Detection and not just
 - Policy Templates

For more information see:

Mark Burgess' Computer Immunology Site

<http://www.iu.hio.no/~mark/research/immune/>

Cfengine Trust Model

- Trusts the local input files
- Trusts the security/authenticity of remote file/policy server
 - strong authentication
 - encrypted data transfer

Automating Security Policy Implementation

19

When considering cfengine's trust model, keep in mind the following questions:

- Who owns the configuration file(s)?
- Who can modify the file(s)?

If the file(s) are modified by the wrong person, does this indicate a purposeful security breach or merely a security hole?

Network Communication

Cfengine uses asymmetric encryption for all network communications. This allows hosts to verify that they are communicating with the host they intend to.

Policy Server Installation

1. Install Cfengine
2. Get `cfserverd` running
 - requires `cfserverd.conf`
3. Configure `cfagent` to communicate with `cfserverd` from step 2.
 - create simple `update.conf/cfagent.conf`
4. Run `cfagent` ever hour
 - `0,30 * * * * /usr/sbin/cfexecd -F`

Client Installation

1. Install Cfengine
2. Configure **cfagent** to communicate with **cfserverd** on your Cfengine Policy Server.
 - create simple **update.conf**
3. Run **cfagent** ever hour


```
0,30 * * * * /usr/sbin/cfexecd -F
```

Automating Security Policy Implementation

21

```
%post
#!/bin/sh
/usr/bin/chvt 3

FILESERVER=192.168.5.6
REMOTEDIR=/home/fedora/4/data
LOCALDIR=/tmp/fedora/

mkdir -p ${LOCALDIR}
mount -o ro,nolock,tcp ${FILESERVER}:${REMOTEDIR} ${LOCALDIR}

echo "* Installing Cfengine"
/usr/bin/yum -y install cfengine
/sbin/chkconfig cfserverd on
/sbin/chkconfig cfexecd on
cp ${LOCALDIR}/*.conf /var/cfengine/inputs/

cat <<'EOF' > /etc/rc.d/rc.local
#!/bin/sh
/usr/sbin/cfagent --no-splay --define sys_startup &
touch /var/lock/subsys/local
EOF
chmod 755 /etc/rc.d/rc.local

echo "Installation complete system will reboot in a few seconds..."
/bin/sleep 90
```

Cfengine Configuration

- Server Configuration
 - `cfserverd.conf`
- Configuration updates
 - `update.conf`
- System updates
 - `cfagent.conf`

Automating Security Policy Implementation

22

Version Control for Configuration Files

It is a good idea to use some versioning control systems such as RCS, CVS, or Subversion. RCS doesn't allow for shared access. For most purposes this is considered a weakness of RCS when compared to other versioning systems; but, for the purposes of system configuration, this may actually be desired.

For more information see:

Using versioning for your configuration files

By: Radu Fericean

<http://www.linux.com/article.pl?sid=06/03/02/174227>

Storing CFEngine configuration in CVS

<http://sial.org/howto/cfengine/repository/>

Using Cfengine with CVS

http://cfwiki.org/cfwiki/index.php/Using_Cfengine_with_CVS

Cfengine Configuration Files

- **Composed of objects**

```
action-type:
  classes-of-host-this-applies-to::
    Actual rule 1
    Actual rule 2 ...
```

- **Example**

```
control:
  actionsequence = ( tidy )
tidy:
  linux::
    /tmp pattern=* age=3 recurse=inf
```

Automating Security Policy Implementation

23

Actions

<code>control</code>	Set up variables; define defaults; determine execution order of the remaining actions.
<code>import</code>	Include objects from another file; used for modularization.
<code>groups</code>	Define classes.
<code>tidy</code>	Delete files.
<code>copy</code>	Define classes.
<code>processes</code>	Test for their existence, signal them, and restart them.
<code>editfiles</code>	Edit ASCII text files.
<code>files</code>	Touch files; check (change) existence, permissions, and ownership;
etc.	
<code>disable</code>	Renames files so that they will not be used.
<code>filters</code>	Select (or prune) during a search over files or processes.
<code>shellcommands</code>	Runs shell commands or scripts.

Classes

- OS architecture (e.g., solaris, linux, NT, ...)
- Unqualified name of a particular host
- User-defined group of hosts
- Day of the week (Monday, Tuesday, ...)
- Hour of the day (Hr00, Hr01, ...)
- Minutes in the hour (Min00, Min01, ...)
- Day of the month (Day1, Day2, ...)
- Month of the year (January, February, ...)

cfservd.conf

```
control:
  domain = ( berkeley.edu )
  AllowUsers = ( root )
  cfrunCommand = ( "/var/cfagent/bin/cfagent" )
admit:
  /masterfiles/inputs *.berkeley.edu
  /var/cfagent/bin/cfagent *.berkeley.edu
```

update.conf

```
control:
  actionsequence = ( copy tidy )
  domain         = ( berkeley.edu )
  policyhost     = ( cfengine.berkeley.edu )
  master_cfinput = ( /masterfiles/inputs )
  workdir        = ( /var/cfengine )
copy:
  $(master_cfinput)  dest=$(workdir)/inputs
                    r=inf
                    mode=700
                    type=binary
                    server=$(policyfile)
tidy:
  $(workdir)/outputs pattern=* age=7
```

Automating Security Policy Implementation

25

```
control:
  actionsequence = ( copy tidy )
  domain         = ( berkeley.edu )
  workdir        = ( /var/cfengine )
  policyhost     = ( cfengine.berkeley.edu )
  master_cfinput = ( /masterfiles/inputs )
  cf_install_dir = ( /usr/sbin )
copy:
  $(master_cfinput)  dest=$(workdir)/inputs
                    r=inf
                    mode=644
                    type=binary
                    exclude=*.lst
                    exclude=*~
                    exclude=#*
                    server=$(policyhost)
  $(cf_install_dir)/cfagent  dest=$(workdir)/bin/cfagent
                             mode=755
                             type=checksum
  $(cf_install_dir)/cfservd  dest=$(workdir)/bin/cfservd
                             mode=755
                             type=checksum
  $(cf_install_dir)/cfexecd  dest=$(workdir)/bin/cfexecd
                             mode=755
                             type=checksum
tidy:
  $(workdir)/outputs pattern=* age=7
```

cfagent.conf

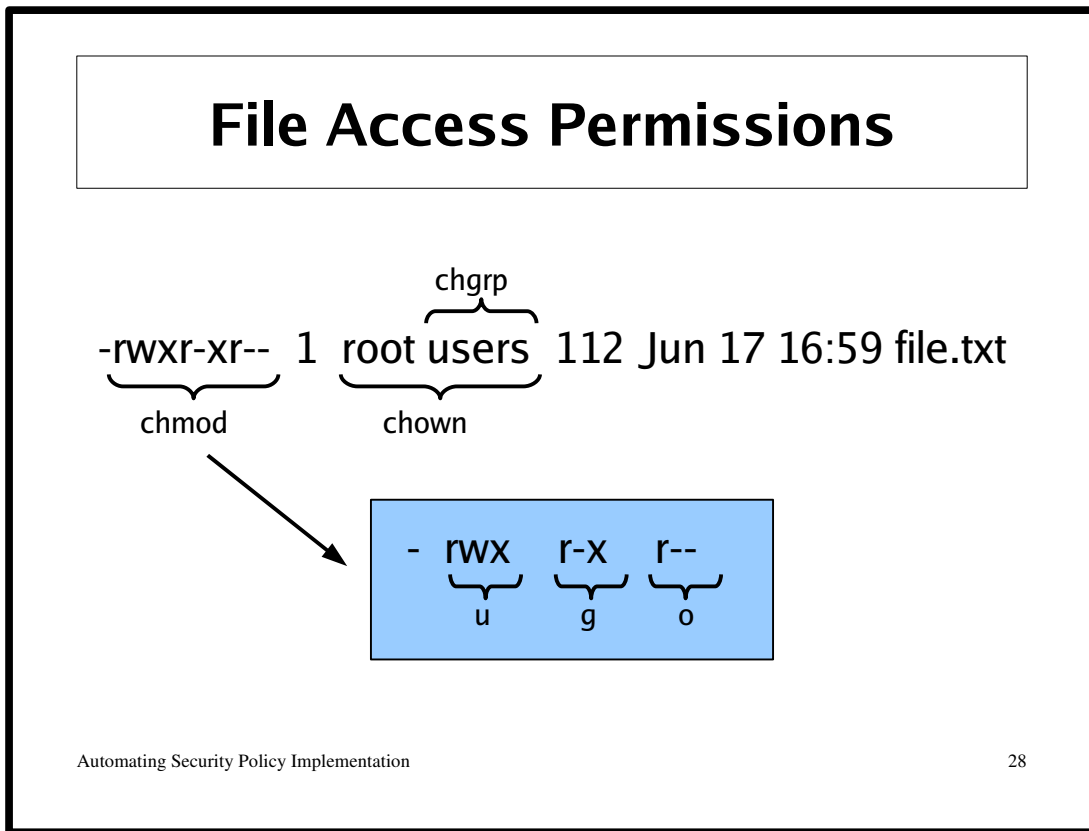
```
control:
  actionsequence = ( processes editfiles )
  domain         = ( berkeley.edu )
  access        = ( root )
  smtpserver    = ( mail.berkeley.edu )
  sysadm       = ( root@berkeley.edu )
processes:
  "cfservd" restart "/usr/sbin/cfservd"
  "cfexecd" restart "/usr/sbin/cfexecd"
editfiles:
  { /etc/crontab
    AppendIfNoSuchLine
    "0 * * * * root /usr/sbin/cfexecd -F"
  }
```

Suspicious Filenames

control:

```
# Security checks
NonAlphaNumFiles = ( on )
FileExtensions = ( o a c gif jpg html ) # etc
SuspiciousNames = ( .mo lrk3 lkr3 )

# Check mail spool directory
WarnNonOwnerMail = ( true )
WarnNonUserMail = ( true )
```



Understanding File Permissions

To see the file permission of a file use the `ls -l` command.

Monitoring Files

```
files:  
  classes::  
    /file-object  
      mode=mode  
      owner=uid-list  
      group=gid-list  
      action=fixall/warnall..  
      ignore=pattern  
      include=pattern  
      exclude=pattern  
      checksum=md5  
      syslog=true/on/false/off
```

File Integrity

```
control:
  ChecksumDatabase = ( /var/cfengine/cache.db )
  ChecksumUpdates = ( false )
files:
  /filename checksum=md5 ....
  /dirname checksum=md5 recurse=inf....
  /var/cfengine/cache.db mode=600 owner=root \
    action=fixall
```

Cfengine provides Tripwire functionality.

Deploying sudoers File

```
editfiles:
{ /etc/sudoers
  AppendIfNoSuchLine
  "jarrod    ALL=(ALL) ALL"
}

files:
/usr/bin/sudo owner=root group=root mode=4111
checksum=md5 action=fixall
/etc/sudoers owner=root group=root mode=0440
action=fixall
```

Automating Security Policy Implementation

31

Sudo

```
less /etc/sudoers
```

Add user permissions

```
su -
visudo
```

Sudo Logging

```
/var/log/secure
sudo /usr/sbin/logwatch -print -service sudo -range all
```

Restrict Root Logins

```
/etc/securetty
```

Special Permissions

- Executables
 - suid (set-user-identifier)
 - sgid (set-group-identifier)
- Directories
 - sgid (set-group-identifier)
 - sticky bit

setuid Root Files

```

filters:
  { root_owned_files
    Owner:      "root"
    Result:     "Owner"
  }
files:
  / filter=root_owned_files mode=u-s
  recurse=inf action=fixall inform=true
  # full paths of files that *should* be SUID
  ignore=/proc ignore=/bin/ping
  ignore=/usr/bin/su ignore=/usr/bin/passwd
  ignore=/usr/bin/crontab
  ...

```

Automating Security Policy Implementation

33

GNU's `find` is your friend!

SUID/SGID

```

find / -xdev -type f -perm +ug=s
find / -perm +4000 -user root

```

Device Special Files

```

find / \( -type b -o -type c \) -ls
find /dev -type f ! -name MAKEDEV

```

World Writable Files

```

find / -path /proc -prune -o \
  -perm +o=w ! \( -type d -perm +o=t \) ! -type l

```

For more information see:

O'Reilly's Linux Security Cookbook

Passwords

- Protect your passwords
- Use hard-to-guess passwords

Do

- * Use a password with mixed-case alphabetic.
- * Use a password with nonalphabetic characters, e.g., digits or punctuation.
- * Use a password that is easy to remember, so you don't have to write it down.
- * Use a password that you can type quickly, without having to look at the keyboard.
- * Use different passwords for account, e.g., your email account, bank account.

Don't

- * Use any words contained in (English or foreign language) dictionaries, spelling lists, or other lists of words in any form (as-is, reversed, capitalized, doubled, etc.)
- * Use information easily obtained about you. E.g., your name, login name, spouse's or child's name, license plate numbers, telephone numbers, social security numbers, the brand of your automobile, the name of the street you live on, etc. use a password of all digits, or all the same letter. This significantly decreases the search time for an intruder.
- * Use a password shorter than 8 characters.
- * Use a keyboard pattern such as qwertyui or oeuidhtn (look at a Dvorak keyboard).
- * Give or share your password, in particular to someone claiming to be from computer support or a vendor.
- * Store or send your password in plain text.
- * Email your password to yourself or someone else.

Deploying Root Passwords

- Add the following to `cfagent.conf`

```
import:
  password.conf
```

- Create `password.conf`

```
editfiles:
  fedora::
    { /etc/shadow
      Backup "off"
      ReplaceFirst "^root\:.*\:" With "root:<...>"
      EndGroup
    }
```

Automating Security Policy Implementation

35

Ideas for creating memorable passwords

- * Use the first letters of the words of a unique phrase or sentence. Then substitute in some digits and symbols. For example, the sentence
I find Jill to be easy-going and straightforward
might yield
1fJ2be-g&sf
Of course, this password should still conform to the above rules.
- * Try combining two or more words together or taking the first (or second or last) letter of each word in an easily remembered phrase. Then mangle the result by adding capitals, digits and punctuation characters. For example,
gOt%L0st! - got lost!
heLP4me\$ - help for me (money)
- * Use misspelled words (WhutdooUmeenIkan'tSpel?).
- * Something that no one but you would ever think of. The best password is one that is totally random to anyone else except you. It is difficult to tell you how to come up with these, but people are able to do it. Use your imagination!

How Do Blackhats Get Passwords?

- Password Crack
 - John the Ripper
- Sniffing
 - tcpdump, ethereal, dsniff, ngrep
- Defaults

Automating Security Policy Implementation

36

“John the Ripper is a fast password cracker, currently available for many flavors of Unix (11 are officially supported, not counting different architectures), DOS, Win32, BeOS, and OpenVMS. Its primary purpose is to detect weak Unix passwords. Besides several crypt(3) password hash types most commonly found on various Unix flavors, supported out of the box are Kerberos AFS and Windows NT/2000/XP/2003 LM hashes, plus several more with contributed patches.”

<http://www.openwall.com/john/>

Wordlists:

<http://www.openwall.com/wordlists/>

<ftp://ftp.ox.ac.uk/pub/wordlists/>

PAM

copy:

```
$(cf_install_dir)/system-auth
  dest=/etc/pam.d/system-auth
  mode=644
  type=checksum
```

Automating Security Policy Implementation

37

```

#%PAM-1.0 /etc/pam.d/system-auth
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth      required      /lib/security/$ISA/pam_env.so
auth      sufficient    /lib/security/$ISA/pam_unix.so \
           likeauth nullok
auth      required      /lib/security/$ISA/pam_deny.so

account   required      /lib/security/$ISA/pam_unix.so
account   sufficient    /lib/security/$ISA/pam_succeed_if.so \
           uid < 100 quiet
account   required      /lib/security/$ISA/pam_permit.so

password  requisite      /lib/security/$ISA/pam_cracklib.so \
           retry=3
password  sufficient    /lib/security/$ISA/pam_unix.so nullok \
           use_authok md5 shadow nis
password  required      /lib/security/$ISA/pam_deny.so

session   required      /lib/security/$ISA/pam_limits.so
session   required      /lib/security/$ISA/pam_unix.so

```

Processes

```
processes:
    "sshd" restart "/usr/sbin/sshd" useshell=false

linux::
    "xinetd" signal=hup restart
        "/usr/sbin/xinetd" useshell=false
    "xntp" restart
        "/usr/sbin/xntpd" useshell=false
    "cron" matches=>1 restart
        "/etc/init.d/cron start" useshell=dumb
    "snmpd" signal=kill
```

Automating Security Policy Implementation

38

Process Monitoring

1. Ensure certain processes are running
2. Ensure some processes are NOT running
3. Send HUP signals to force configuration updates

Processes

- Viewing (`ps`, `pstree`, `top`)
- Signalling (`kill`, `killall`)
- Accounting (`psacc`)
- Investigating (`lsof`, `/proc/[0-9]*`)

Service Monitoring

- **Disabling telnet**

```
editfiles:
  { /etc/xinetd.d/telnet
    ReplaceFirst "[\s\t]?disable[\s\t]?=" With
      "          disable          = yes"
    DefineClasses "modified_xinetd"
  }
processes:
  modified_xinetd::
    "xinetd" signal=hup
```

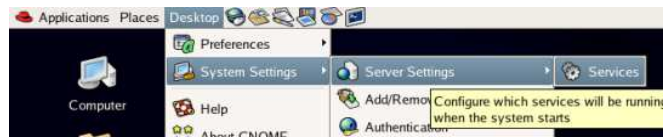
Automating Security Policy Implementation

39

Inetd Example

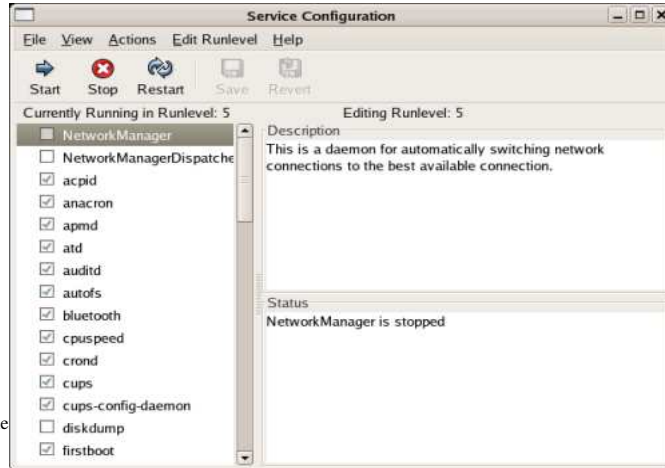
```
editfiles:
  { /etc/inetd.conf
    HashCommentLinesContaining "rshd"
    HashCommentLinesContaining "rlogind"
    DefineClasses "modified_inetd"
  }
processes:
  modified_inetd::
    "inetd" signal=hup
disable:
  /root/.rhosts
  /etc/hosts.equiv
```

Services



- `chkconfig`
- `service`

Automating Security Policy Implementation



`system-config-services`

Service Monitoring, Revisited

- Ensure `httpd` is running

```
processes:
```

```
  http_server::
```

```
    "httpd" restart "/sbin/service httpd restart"  
    elsedefine=httpd_chkconfig_on
```

Service Monitoring, Revisited

- Ensure `httpd` will start when changing runlevel

shellcommands:

```
http_server.httpd_chkconfig_on.fedora::  
    "/sbin/chkconfig httpd on" useshell=false
```

Automating Security Policy Implementation

42

Runlevels

Specifies the set of services that should be running. Runlevels are associated with a number from 1-6:

- 0: Halt
- 1: Single User Mode
- 2: Basic Multi-user Mode
- 3: Full Multi-user without X
- 4: Not Used
- 5: Mutli-user with X
- 6: Reboot

Default runlevel is specified in `/etc/inittab`. You can force a system into a specific runlevel with the `init` command. The specific services that are to be started or killed when going from one runlevel to another are found in `/etc/rc.d`.

Service Monitoring, Revisited

- Disable `httpd` on some machines

```
processes:
  !http_server::
    "httpd"
    matches=0
    action=bymatch
    define=httpd_chkconfig_off

shellcommands:
  !http_server.fedora::
    "/sbin/chkconfig --list http | grep on" useshell=true
    inform=false
    define=httpd_chkconfig_off
  !http_server.httpd_chkconfig_off.fedora::
    "/sbin/chkconfig httpd off" useshell=false
```

Service Monitoring, Revisited

- **Alternatively**

```
shellcommands:  
  "/sbin/service httpd status > /dev/null"  
  define=httpd_service_is_on  
  "/sbin/chkconfig httpd"  
  define=httpd_chkconfig_is_on  
http_server.!httpd_service_is_on::  
  "/sbin/service httpd start"  
http_server.!httpd_chkconfig_is_on::  
  "/sbin/chkconfig httpd on"  
!http_server.httpd_service_is_on::  
  "/sbin/service httpd stop"  
!http_server.httpd_chkconfig_is_on::  
  "/sbin/chkconfig httpd off"
```

TCP wrappers

- Maintain a pair of master files on a trusted host
 - /masterfiles/hosts.allow
 - /masterfiles/hosts.deny
- Distribute to each host by cfengine
 - /etc/hosts.allow
 - /etc/hosts.deny

Automating Security Policy Implementation

45

The TCP wrappers package

host-based access control to network services

/usr/lib/libwrap.a library

access control and logging the name of the requesting host and the requested service

most network services within Fedora Linux are linked against the libwrap.a library

Example

copy:

```
/masterfiles/hosts.deny          dest=/etc/hosts.deny
                                  mode=644
                                  server=trusted
/masterfiles/hosts.allow         dest=/etc/hosts.allow
                                  mode=644
                                  server=trusted
```

Fedora Package Management

- RPM Querying
- RPM File Monitoring
- Signature Verification

Cfengine Package Management

- Update openssh

```
control:
  <...>
  DefaultPkgMgr = ( rpm )

packages:
  openssh version=4.2p1-fc4.10 cmp=ge
  elsedefine=updatessh

shellcommands:
  updatessh::
    "/usr/bin/yum -y install openssh"
    useshell=false
```

Automating Security Policy Implementation

47

ARPMATS Project

This project provides a convenient way to manage which applications are installed on which machines.

<http://arpmats.sourceforge.net/>

Looking ahead: Cfengine 3

- Complete rewrite
- Move from C to C++
- Based on Promise Theory

Automating Security Policy Implementation

48

Cfengine 3 will be a complete rewrite of the language, parser, and will include:

- Reentrant locks, allowing finer recursion control.
- Intelligent list variables `x = List(a b c d, " ")`
- C++ object methods for constructors
- A generalized language design, that is more template configurable
- Generalized interface to actions - for possible multiple APIs.
- Use of "inheritance" where possible for greater flexibility through templates and filters.
- Priority evaluation concepts

Service Level Agreements and Quality of Service concepts.

See:

<http://www.cfengine.org/develop.phtml?page=plan.txt>