



DSci526: Secure Systems Administration

Linux Administration
Accreditation and Acceptance

Prof. Clifford Neuman

Lecture 12
14 April 2021
Online



Announcement

- Alternate Schedule for April 21st Lecture
 - Wednesday 21 April
 - 10:30AM to 1:50PM
 - Same Zoom Link



Agenda

1405-1535 Linux Administration

Alejandro Najera	->	Administration
Tejas Pandey	->	Identity Management
Ayush Ambastha	->	Kernel Security
Azzam Alaseed	->	SELinux

1535-1545 Break

1545-1645

1645-1650 General Class Discussion of Second Group Project

1650-1655 Second Project Briefing by Team 1

1655-1700 Second Project Briefing by Team 2

1700 Breakouts for Second Group Project



Linux Presentation

Alejandro Najera ->
Tejas Pandey ->
Ayush Ambastha ->
Azzam Alaseed ->

Administration
Identity Management
Kernel Security
SELinux



Linux System Administration

Alejandro Najera

The Linux System Administrator

Responsibilities & Common Duties:



- Auditing
 - Logging
- Monitoring
 - Host & Network
- High Availability
 - Clustering



Auditing & Logging

Critical security component of any system:

Goals of Auditing:

- Compliance
- Security
- Forensics

Types of Logs:

- Operating System
- Application
- Storage & Database
- Infrastructure
- Administration



Auditing & Logging in Linux

Default log location on Host: /var/log

Centralized Logging:

- Syslog
- Syslog-ng
- Rsyslog

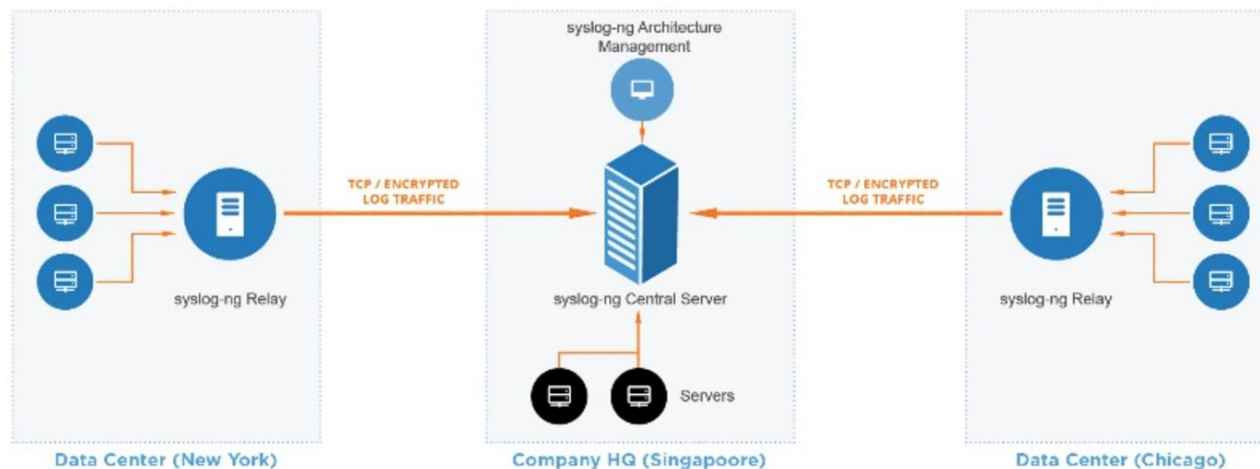
Host Logs:

- wtmp
- utmp
- dmesg
- messages
- maillog or mail.log
- spooler
- auth.log or secure

Auditing & Logging in Linux

SCALING SYSLOG-NG

- Client – Relay – Server instead of Client – Server
- Distribute some of the processing to Client/Relay





Monitoring

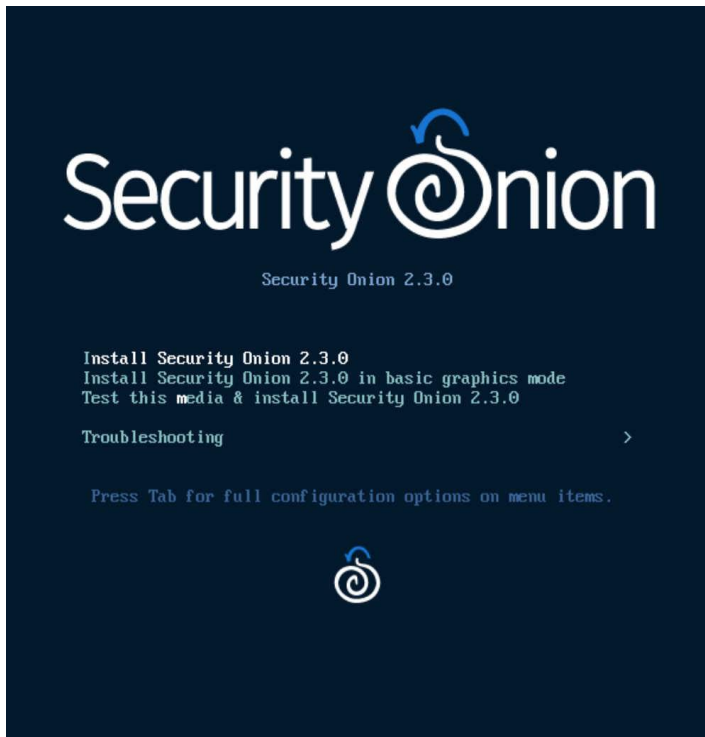
Detect and Respond to Cyber Threats:

Goals of Monitoring:

- Survey
- Audit
- Assess
- Host Monitoring
 - Removable Media
 - File Integrity
 - Malware
- Network Network
 - Intrusion Detection
 - Intrusion Prevention
 - SIEM
 - Asset Management

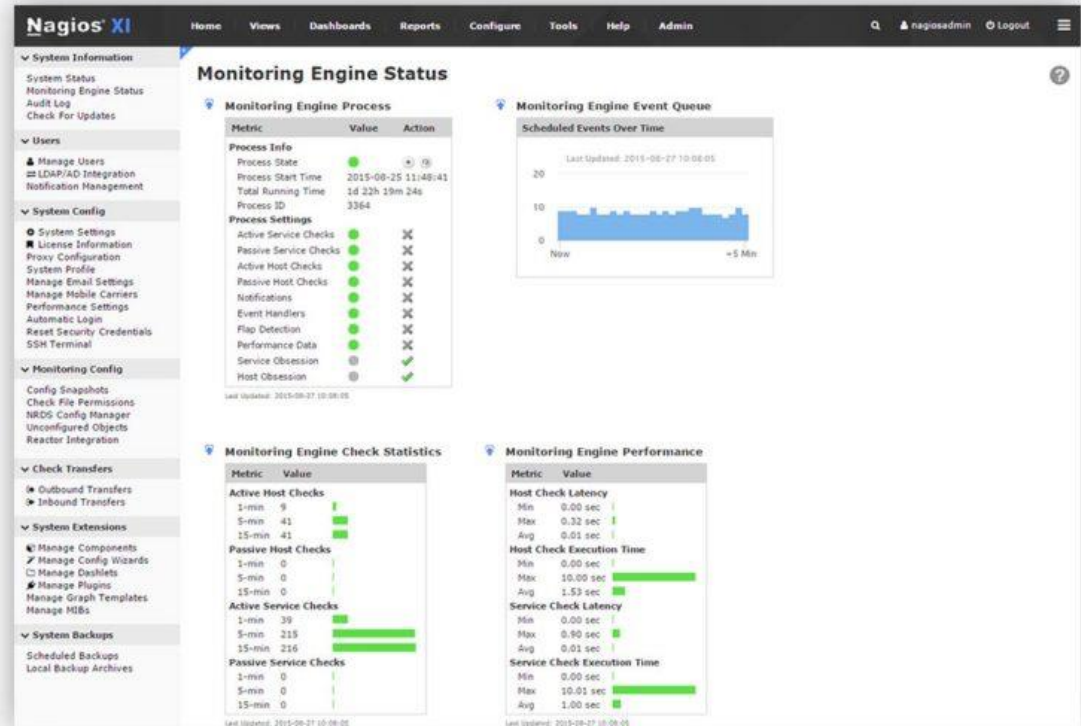
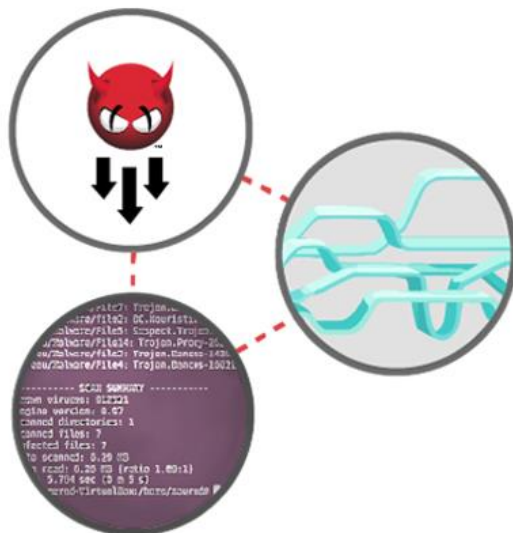
Monitoring in Linux

Open Source Tools :



- Nagios
 - Monitors systems, networks & infrastructure
- ClamAV
 - Open-Source AntiVirus
- USBGuard
 - Removable Media Policy & Enforcement
- OSSEC
 - HIDS, File Integrity

Monitoring in Linux





High Availability

Reduce the impact of service interruptions:

Goals of High Availability:

- Eliminate single point of failure
- Provide continued service when system components fail.

Components:

- Backups
- Load balancing



- Redundancy

High Availability in Linux

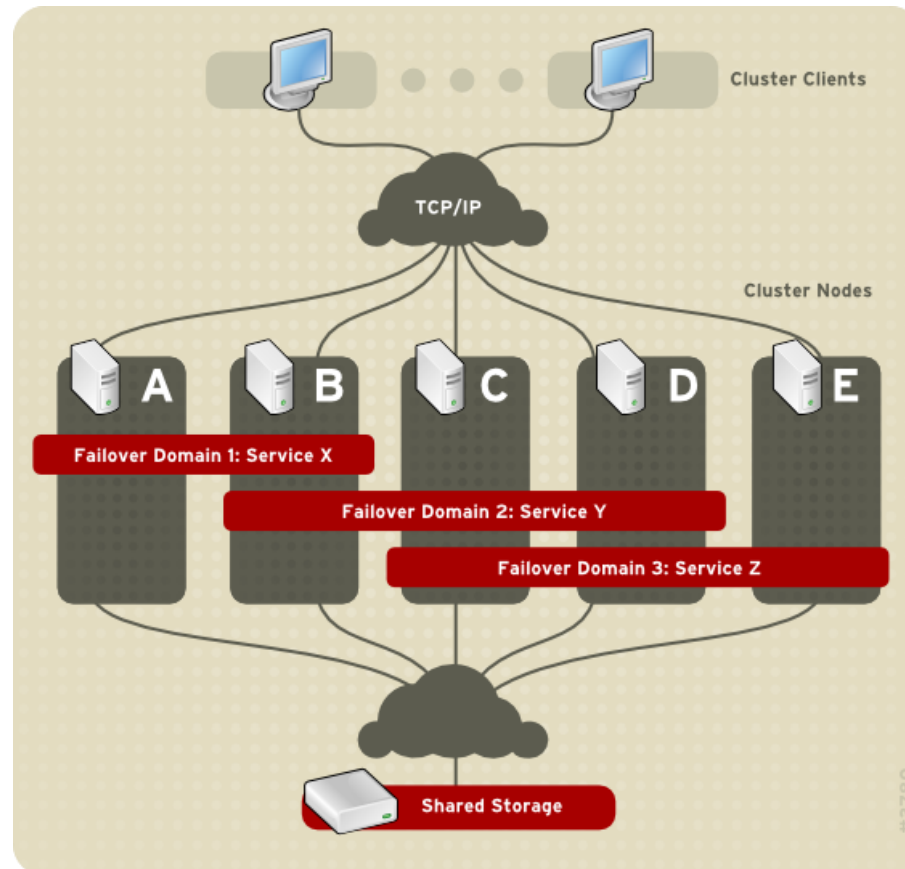
Tools & Methods:



The screenshot shows the Red Hat Cluster and Storage Systems web interface. The top navigation bar includes the Red Hat logo and the text "CLUSTER AND STORAGE SYSTEMS". Below the navigation bar, there are tabs for "homebase", "cluster", and "storage". The "cluster" tab is selected. On the left side, there is a sidebar with a "clusters" section containing links for "Cluster List", "Create a New Cluster", and "Configure". Below this, there is a section for "my_rh_cluster" with links for "Nodes", "Services", "Resources", "Failover", "Domains", "Shared Fence Devices", "Add a Fence Device", and "Configure a Fence Device". The "Add a Fence Device" link is highlighted. The main content area shows the configuration for "my_rh_cluster" under the heading "Add a Sharable Fence Device". It includes a "Fencing Type" dropdown menu set to "APC Power Switch". Below this, there are input fields for "Name", "IP Address", "Login", and "Password". At the bottom, there is a button labeled "Add this shared fence device".

- Amanda backup & recovery
- Clustering w/ Corosync & Pacemaker
- Fencing

High Availability in Linux





References (Linux Administration)

- › http://www.linuxtopia.org/online_books/linux_administrators_security_guide/
- › <https://access.redhat.com/documentation/en-us/>
- › <https://learning.oreilly.com/api/v1/dashboard/continue/linuxbasicsforhackers/>
- › <https://learning.oreilly.com/api/v1/continue/unixandlinuxsystemadministrationhandbook/>
- › <https://securityonionsolutions.com/software/>



Identity Management in Linux

Tejas Pandey

What is Identity Management?



- The practice of managing access to enterprise resources to keep systems and data secure
- Is one of the key component in security architecture, as it can help verify users' identities before granting them the right level of access to resources

Challenges



- Problem with traditional identity management (IdM) approaches:
 - Password Fatigue - move to SaaS and cloud-based microservices, each application with own identity store and password requirements for login. User has to manage multiple identities, diminishing productivity, risk of reuse.
 - Failure prone onboarding and offboarding process - accounts managed at department level, access granted by application administrator
 - Where are users stored?
 - What properties/attributes do they have?
 - How can services and systems access this data?

18

Challenges



- Compliance and audit - no clear visibility in terms of user access, audit trail and security policies.

Problem Space



- Main aspects of an IdM solution:
 - Centralized Management of Identities/Security Policies
 - Provide Various Authentication/Authorization Mechanisms
 - Enterprise Single Sign-On
 - Compliance and Audit

Benefits of Identity Management



- Enterprise single sign-on
 - Without IdM:
 - Users log in to the system and are prompted for a password every single time they access a service or application. These passwords might be different, and the users have to remember which credential to use for which application.
 - With IdM: After users log in to the system, they can access multiple services and applications without being repeatedly asked for their credentials.
 - Improve usability, increase productivity
 - Reduce the security risk of passwords being written down or stored insecurely

21

Benefits of Identity Management



- Managing identities and policies with several Linux servers
 - Without IdM
 - Each server is administered separately. All passwords are saved on the local machines. The IT administrator manages users on every machine, sets authentication and authorization policies separately, and maintains local passwords
 - With IdM
 - Maintain the identities in one central place: the IdM server
 - Apply policies uniformly to multiples of machines at the same time
 - Set different access levels for users by using host-based access control, delegation, and other rules
 - Centrally manage privilege escalation rules

22

The FreeIPA Project



- The **I**ntity **P**olicy and **A**udit management solution
 - Upstream version of RedHat's Identity Management offering
 - Combines LDAP, Kerberos, DNS and certificate management capabilities
 - Provides centralized authentication, authorization and identity information for Linux/UNIX infrastructure
 - Enables centralized policy and privilege escalation management
 - Supports integration with Microsoft's Active Directory and Cross-realm Kerberos Trust/Authentication

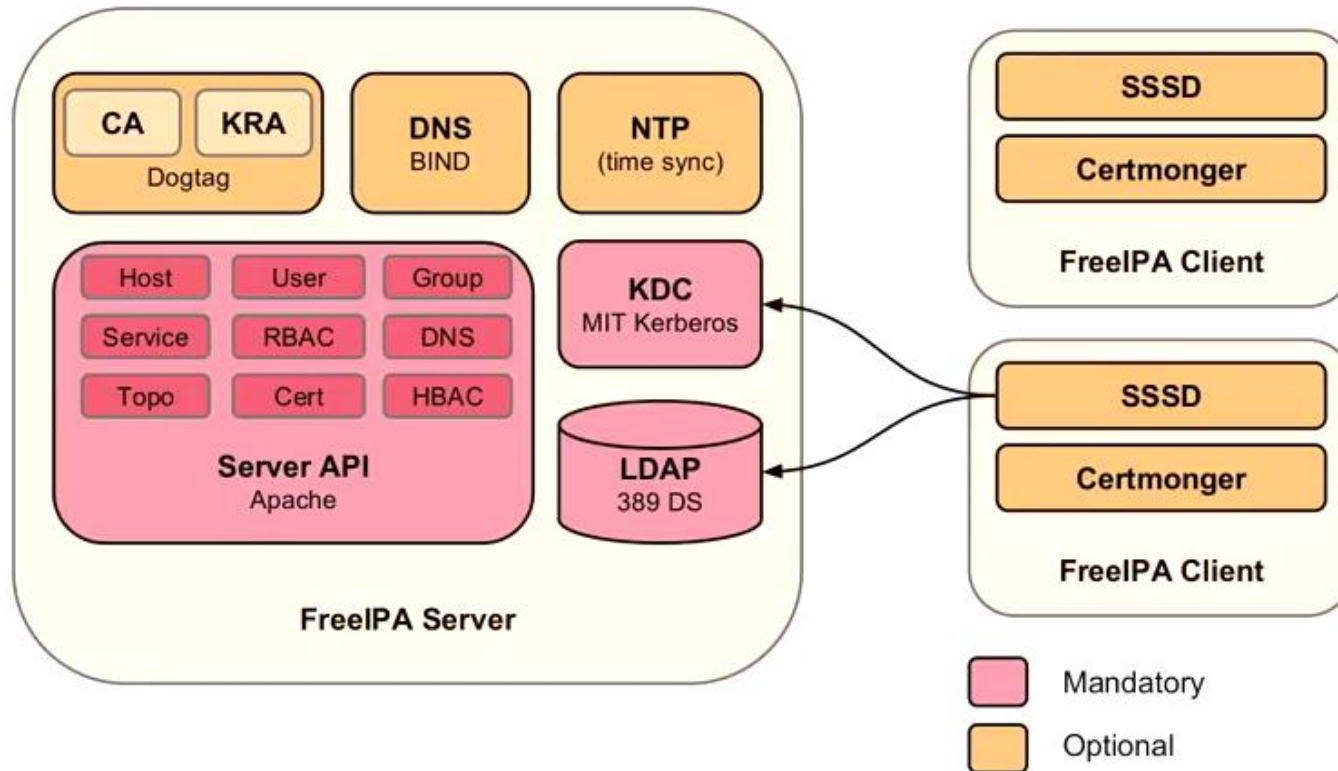
FreeIPA vs Standard LDAP Implementation



FreeIPA	LDAP Directory Server
Has a specific purpose: managing identities as well as authentication and authorization policies that relate to these identities	A general-purpose directory; it can be customized to fit a broad range of use cases
Schema - a specific schema that defines a particular set of entries relevant to its purpose, such as entries for user or machine identities	Schema - a flexible schema that can be customized for a vast array of entries, such as users, machines, network entities, physical equipment, or buildings
Typical usage - the identity and authentication server to manage identities within the boundaries of an enterprise or a project.	Typical usage - a back-end directory to store data for other applications, such as business applications that provide services on the Internet.



FreeIPA Architecture



Source:

25

Authentication

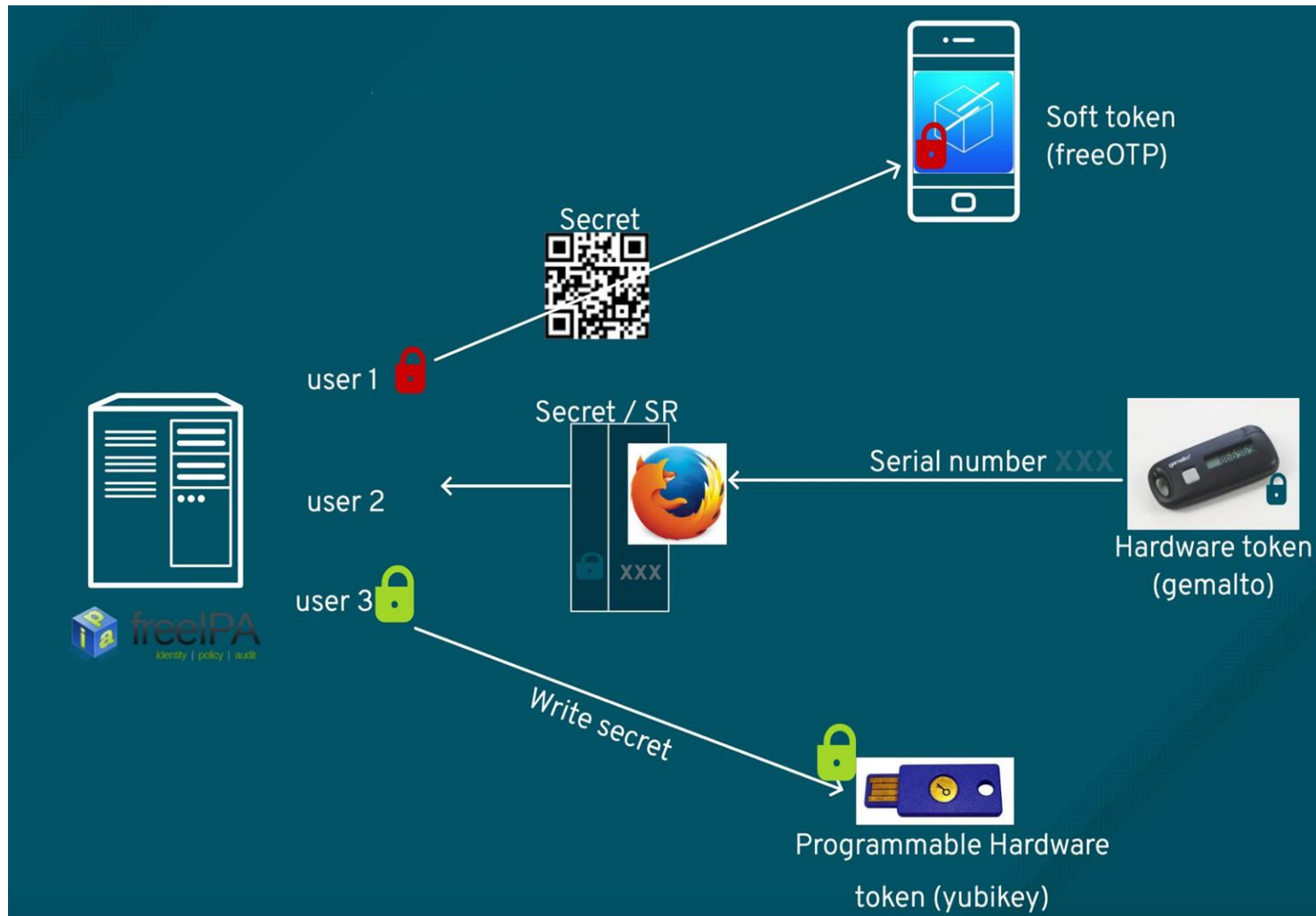


- FreelPA offers variety of user authentication mechanisms, mainly:

- Password
- 2FA, Password + OTP (TOTP and HOTP supported)
- RADIUS User authentication types ⓘ

- ☐ Password
- ☐ RADIUS
- ☐ Two factor authentication (password + OTP)
- ☐ PKINIT
- ☐ Hardened Password (by SPAKE or FAST)

Multi Factor Authentication



Multi Factor Authentication



- Supported:
 - Yubikey (HOTP)
 - Software token (TOTP)
 - Smartcards

```
[bash-3.2$ ssh tpandey@104.200.24.192
```

```
[First Factor:
```

```
[Second Factor:
```

```
Last login: Wed Apr 14 00:05:20 2021 from 2601:647:4000:
```

```
[tpandey@li773-192 ~]$ █
```

Add OTP token

Type

☒ Time-based (TOTP) ☐ Counter-based (HOTP)

Unique ID

Description

Owner

Validity start

YYYY-MM-DD

hh

:

mn

UTC

Validity end

YYYY-MM-DD

hh

:

mn

UTC

Vendor

Model

Serial

Key

Algorithm

☒ sha1 ☐ sha256 ☐ sha384 ☐ sha512

Digits

☒ 6 ☐ 8

Clock interval
(seconds)

* Required field

Add

Add and Add Another

Add and Edit

Cancel

Access Control



- FreeIPA offers access control management in the form of:
 - SUDO policy
 - Role based access control (via SUDO)
 - Host based access control
 - SELinux user and role mapping

Access Control (SUDO)



- Sudo policy defines:
 - Sudo options - for example, (!authenticate)
 - Who - is allowed to invoke sudo (users, user-groups)
 - Where - a user or user-group is allowed sudo privileges. (hosts/systems)
 - What - commands can be invoked through sudo
 - Support only need to read access to logs (tail | less)
- FreeIPA supports caching of sudo rules via SSSD
 - Needed in situations where client can't access IPA server
 - Caveat - user has to have accessed the system before; can't cache what you don't know

Sample SUDO Policy (CLI)



- Requirement: granting sudo access on certain hosts, on all commands:
 - `ipa group-add db-admins --desc="Database Admins"`
 - `ipa group-add-member db-admins --users=jdoe --users=jsmith`
 - `ipa sudorule-add --cmdcat=all sudo-db-admins`
 - `ipa sudorule-add-user --groups=db-admins sudo-db-admins`
 - `ipa sudorule-add-host sudo-db-admins --hosts=db1.example.com --hosts=db1.example.com`

Sample SUDO Policy (GUI)



Settings

Reset Update Expand All

▶ ACCESS THIS HOST

▼ RUN COMMANDS

ALLOW

Command category the rule applies to: ☐ Any Command ☒ Specified Commands and Groups

<input type="checkbox"/>	Sudo Commands	X Delete + Add
<input type="checkbox"/>	/usr/bin/less	
<input type="checkbox"/>	/usr/bin/vim	
<input type="checkbox"/>	Sudo Command Groups	X Delete + Add
<input type="checkbox"/>	files	

DENY

<input type="checkbox"/>	Sudo Commands	X Delete + Add
<input type="checkbox"/>	Sudo Command Groups	X Delete + Add

32

Access Control (HBAC)



- HBAC policy defines:
 - Which users or group of users can access
 - Which hosts or groups of hosts
 - Using which login services:
 - ssh, ftp, sftp, telnet etc..
 - Not a replacement for firewall rules!

Sample HBAC Policy



- Requirement: allows users in usergroup *sysadm* to access hosts in hostgroup *webservers*
 - `ipa hbacrule-add sysadmin_webservers`
 - `ipa hbacrule-add-host sysadmin_webservers --hostgroup webservers`
 - `ipa hbacrule-add-user sysadmin_webservers --group sysadmin`
 - Test a HBAC rule:
 - `ipa hbactest --host client.web1.com --service sshd --user bob`

Access Control (SELinux)



- SELinux mappings can be defined centrally.
- Allow different users on different systems have different SELinux context.
- Default SELinux labels are available in FreeIPA configuration.
- Mappings are enforced on the client (SSSD). And cached, if needed.

SELinux User Capabilities



User	Role	Domain	X Window System	su or sudo	Execute in home directory and /tmp (default)	Networking
sysadm_u	sysadm_r	sysadm_t	yes	su and sudo	yes	yes
staff_u	staff_r	staff_t	yes	only sudo	yes	yes
user_u	user_r	user_t	yes	no	yes	yes
guest_u	guest_r	guest_t	no	no	yes	no
xguest_u	xguest_r	xguest_t	yes	no	yes	Firefox only

Audit



- Problem statement:
 - Organizations in highly regulated industries like financial and healthcare must meet audit requirements (accountability) for employees and contractors.
- The FreeIPA Session recording feature aims to answer the following:
 - What user accessed the system?
 - Commands executed while inside the system
 - How can malicious activity be attributed (audit trail)?

Session Recording



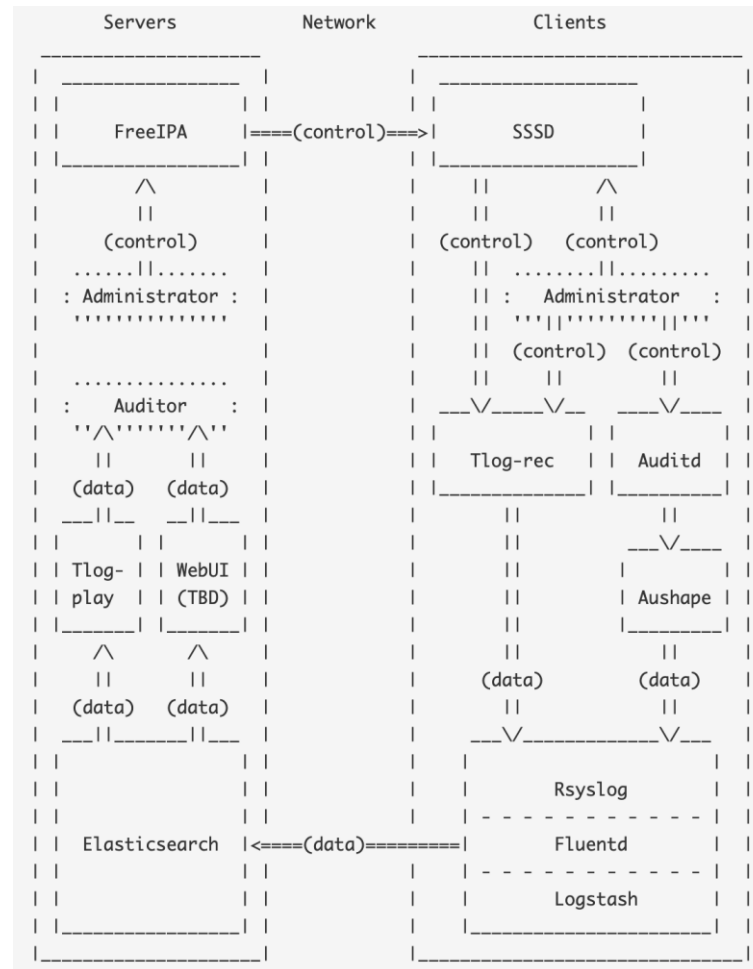
- Client-side components:
 - **Tlog-rec** – recording process, invoked in place of the user's shell when he/she logs into the system. The recording process creates a pseudo-terminal, starts the actual user shell under it, and records everything that passes between the pseudo-terminal and the actual user terminal.
 - **Auditd** – general system auditing subsystem in Linux, which collects all the activity related to user session in the form of the audit entries.
 - **SSSD** – defines users and user groups to enable Session recording for.
 - **Logging server** - Rsyslog, Fluentd, or Logstash – a collection agent, which streams the audit and session recording data from the system to data aggregation server (Elasticsearch).

Session Recording



- Server-side components:
 - **ElasticSearch** - the data storage where the session recording and audit data can be placed and correlated.
 - **Tlog-play** - terminal based playback tool which can be used from the command line to recreate the session.

Session Recording



40

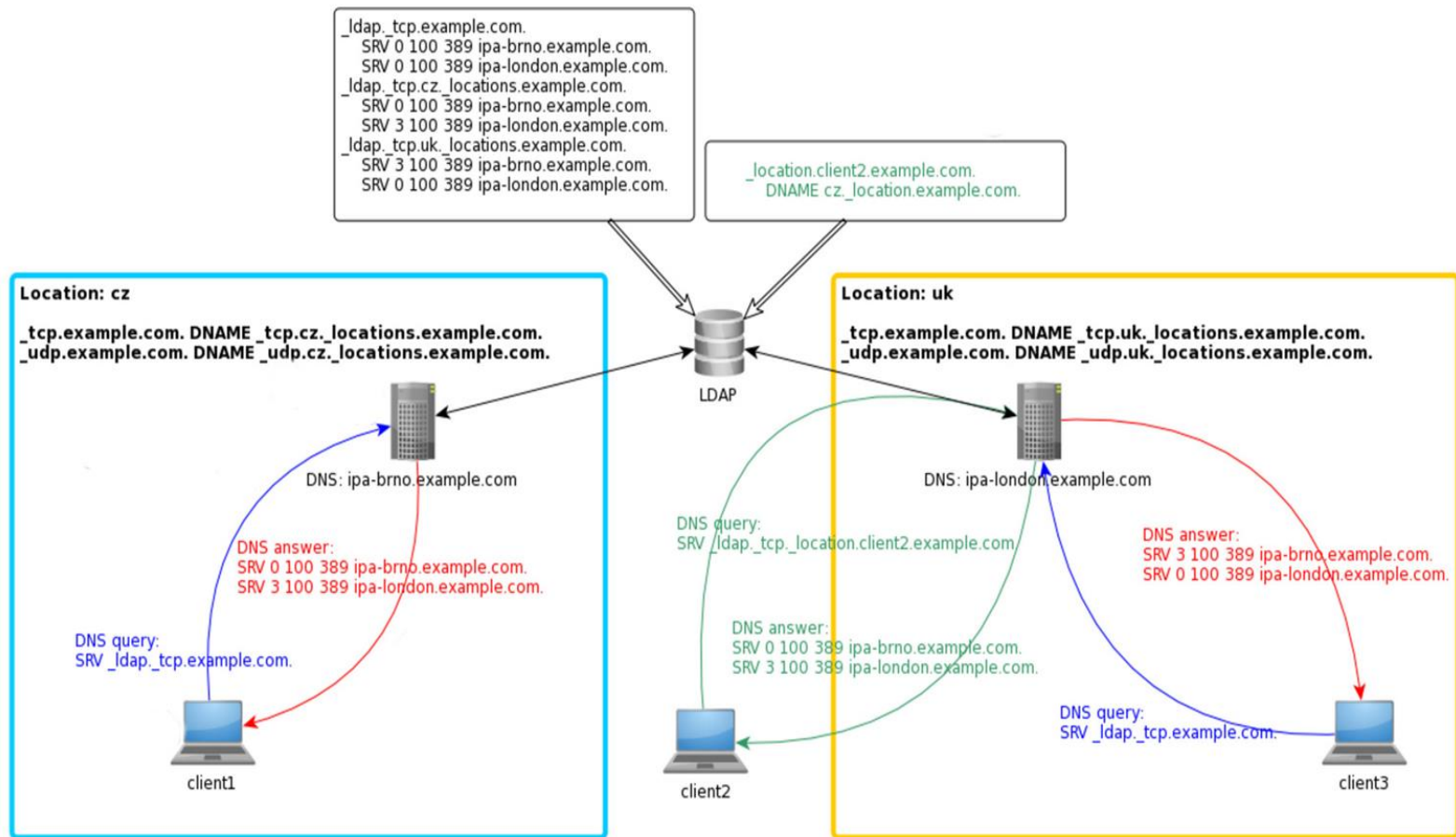
High Availability



- Support for Multi-master replication:
 - Information is shared between FreeIPA servers and replicas. Multi-master means servers and replicas all receive updates and, therefore, are data masters.
- Support for service auto-discovery and location based load balancing:
 - Group replicas based on geolocation
 - Assign priorities to LDAP and Kerberos DNS SRV records
 - Hosts autodiscover services based on SRV records, queries the nearest replica based on SRV record priority.

41

High Availability



42

References



1. <https://freeipa.readthedocs.io/en/latest/index.html>
2. https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/linux_domain_identity_authentication_and_policy_guide/introduction
3. https://www.freeipa.org/page/Directory_Server
4. <https://www.freeipa.org/page/Kerberos>
5. <https://www.freeipa.org/page/PKI>
6. <https://www.freeipa.org/page/DNS>
7. <https://www.freeipa.org/page/Certmonger>
8. https://www.freeipa.org/page/Web_UI
9. <https://www.freeipa.org/page/Trusts>



Linux Kernel Security

Ayush Ambastha



Agenda

- What is a Kernel and why is it important?
- Examples of Security Vulnerabilities
- Various Security Features
- How to Protect Your Linux System
- References



What is a Kernel and why is it important?

- The Linux kernel is the core component of the Linux operating system, maintaining complete control over everything in the system.
- It is the interface between applications and data processing at the hardware level, connecting the system hardware to the application software.
- The kernel manages input/output requests from software, memory, processes, peripherals and security, among many other functions.
- Security of the kernel determines the security of the Linux operating system as a whole, as well as the security of every individual system that runs on Linux.



Examples of Security Vulnerabilities

- **CVE-2017-18017:** This critical vulnerability, which exists in the netfilter `tcpmss_mangle_packet` function, is extremely dangerous because of the important role that it plays in filtering network communications by defining the maximum segment size that is allowed for accepting TCP headers. Without these controls in place, users are susceptible to overflow issues and DoS attacks.
- **CVE-2016-10150:** This use-after-free vulnerability affecting Linux kernel versions prior to 4.8.13 allows users to cause a DoS attack. This flaw could also be exploited by hackers to gain privileges.
- **CVE-2015-8812:** This severe vulnerability impacting versions prior to 4.5, which was discovered in the drivers of the Linux kernel, enables remote attackers to execute arbitrary code or cause a DoS (use-after-free) via crafted packets.



Security Features

- Discretionary Access Control (DAC)
- Extended DAC
 - POSIX ACLs
 - POSIX Capabilities
- Namespaces
- Network Security
- Cryptography
- Linux Security Modules
- Memory Protection
- Audit
- Integrity Management



Discretionary Access Control (DAC)

- The security features of the Linux kernel have evolved significantly to meet modern requirements, although DAC remains as the core security model.
- DAC allows the owner of an object (such as a file) to set the security policy for that object. For example, create a new file in your home directory and decide who else may read or write the file.
- This policy is implemented as permission bits attached to the file's inode.
- Permissions can be set separately for the owner, group and others (i.e. everyone else) to form a basic Access Control List (ACL).
- The root user (superuser) bypasses DAC policy for the purpose of managing the system. Running a program as the root user provides that program with all rights on the system.



POSIX ACLs

- POSIX (Portable Operating System Interface)
- They extend the DAC ACLs to a much finer-grained scheme, allowing separate permissions for individual users and different groups.
- They're managed with the setfacl and getfacl commands and the ACLs are managed on disk via extended attributes.

POSIX Capabilities

- The aim of this feature is to break up the power of the superuser, so that an application requiring some privilege does not get all privileges.
- The application runs with one or more coarse-grained privileges, such as CAP_NET_ADMIN for managing network facilities.
- Capabilities for programs may be managed with the setcap and getcap.
- It's possible to reduce the number of setuid applications on the system by assigning specific capabilities to them.



Namespaces

- Namespaces are a feature of the Linux kernel that partitions kernel resources such that one set of processes sees one set of resources while another set of processes sees a different set of resources.
- This is not primarily a security feature, but is useful for implementing security.
- One example is where each process can be launched with its own, private /tmp directory, invisible to other processes, and which works seamlessly with existing application code. This eliminates the a lot of security threats that existed because of shared folders among processes.
- Linux Namespaces have been used to help implement multi-level security, where files are labeled with security classifications, and potentially entirely hidden from users without an appropriate security clearance.



Network Security

- Netfilter is a framework provided by the Linux kernel that allows various networking-related operations to be implemented in the form of customized handlers. Kernel-level modules may hook into this framework to examine packets and make security decisions about them.
- `iptables` is one such module, which implements an IPv4 firewalling scheme, managed via the userland iptables tool. Access control rules for IPv4 and IPv6 packets are installed into the kernel, and each packet must pass these rules to proceed through the networking stack.
- The networking stack also includes an implementation of IPsec, which provides confidentiality and integrity protection of IP networking. It can be used to implement VPNs, and also point to point security.
- Stateful packet inspection and Network Access Translation (NAT) is also implemented in the codebase.



Cryptography

- Crypto API is a cryptography framework in the Linux kernel, for various parts of the kernel that deal with cryptography, such as IPsec and dm-crypt.
- It provides support for a wide range of cryptographic algorithms and operating modes, including commonly deployed ciphers, hash functions, and asymmetric cryptography.

Linux Security Modules

- Linux security module (LSM) is the framework integrated into the kernel to provide the necessary components to implement the Mandatory access control (MAC) modules, without having the need to change the kernel source code every time.



Memory Protection

- Address space layout randomization (ASLR) is a memory-protection process for operating systems that guards against buffer-overflow attacks by randomizing the location where system executables are loaded into memory.
- The success of many cyber attacks, particularly zero-day exploits, relies on the hacker's ability to know or guess the position of processes and functions in memory. ASLR is able to put address space targets in unpredictable locations. If an attacker attempts to exploit an incorrect address space location, the target application will crash, stopping the attack and alerting the system.
- It doesn't resolve vulnerabilities, but makes exploiting them more of a challenge.



Audit

- The Linux Auditing System is a native feature to the Linux kernel that collects certain types of system activity to facilitate incident investigation.
- The Linux Auditing subsystem is capable of monitoring 3 distinct things:
 - System calls: See which system calls were called, along with contextual information like the arguments passed to it, user information, etc.
 - File access: This is an alternative way to monitor file access activity, rather than directly monitoring the open system call and related calls.
 - Select, pre-configured auditable events within the kernel.
- Using these categories of events, you can audit activity like authentications, failed cryptographic operations, abnormal terminations, program execution, and SELinux modifications.



Example of Audit record

```
type=1300 msg=audit(04/10/2021 13:37:89.567:444): arch=c000003e  
syscall=323 success=yes exit=42 a0=feae74 a1=80000 a2=1b6 a3=9434e28  
items=1 ppid=1337 pid=1812 auid=300 uid=500 gid=500 euid=0 suid=0 fsuid=0  
egid=500 sgid=500 fsgid=500 tty=pts2 ses=2 comm="sudo" exe="/usr/bin/sudo"  
subj=unconfined_u:unconfined_r:unconfined_t:20-s0:c0.c1023 key="test_audit"
```

- User with auid 300
- Used an ssh terminal to use the sudo command
- Invoked the userfaultfd syscall as root
- Date - April 10th, 2021 at 13:37 GMT.

You can see how this level of detail can be useful for inspecting system behavior during incident response and building a timeline of activity.

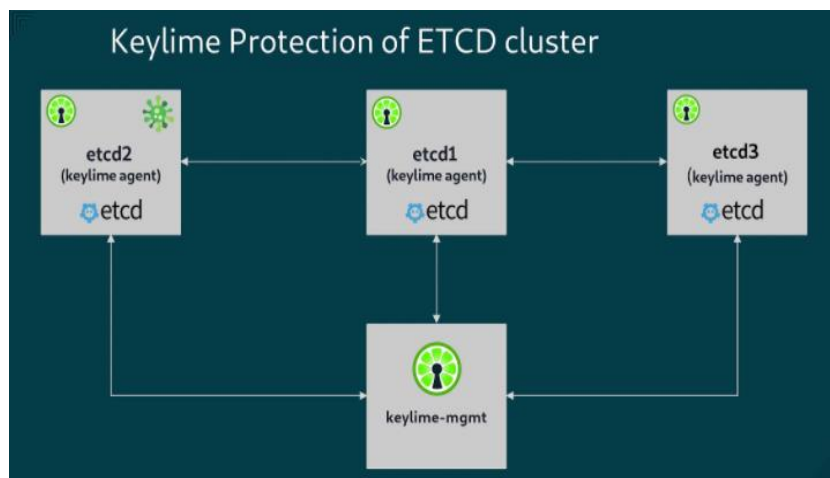


Integrity Management

- The Integrity Measurement Architecture (IMA) component performs runtime integrity measurements of files using cryptographic hashes, comparing them with a list of valid hashes.
- The list itself may be verified via an aggregate hash stored in the TPM. Measurements performed by IMA may be logged via the audit subsystem, and also used for remote attestation, where an external system verifies their correctness.
- IMA may also be used for local integrity enforcement via the Appraisal extension. Valid measured hashes of files are stored as extended attributes with the files, and subsequently checked on access.
- If a file has been modified, IMA may be configured via policy to deny access to the file. The Digital Signature extension allows IMA to verify the authenticity of files in addition to integrity by checking RSA-signed measurement hashes.

Keylime

- Keylime is a CNCF (Cloud Native Computing Foundation) hosted project that provides a highly scalable remote boot attestation and runtime integrity measurement solution. Keylime enables users to monitor remote nodes using a hardware based cryptographic root of trust (TPM).
- With IMA (Integrity Measurement Architecture) - it monitors the runtime environment as the system calls are made. IMA makes a cryptographic hash of the obj and keylime compares that to a whitelist of a trusted states.





How to Protect Your Linux System

- Update your system frequently
- When using a stable version, plan ahead and upgrade to the next version before official support is ended
- Implement proper firewall filtering policies
- Disable direct memory access (DMA) to prevent DMA attacks
- Create regular backups
- Set up system monitoring tools to avoid downtime



References

- <https://www.kernel.org/doc/html/latest/>
- <https://www.linux.com/training-tutorials/overview-linux-kernel-security-features/>
- “How ASLR protects Linux systems from buffer overflow attacks”, Sandra Henry-Stocker, Network World, JAN 8, 2019.
- <https://linuxsecurity.com/features/features/linux-kernel-security-in-a-nutshell-how-to-secure-your-linux-system>
- <https://resources.whitesourcesoftware.com/blog-whitesource/top-10-linux-kernel-vulnerabilities>
- <https://keylime-docs.readthedocs.io/en/latest/index.html>
- <https://github.com/keylime/keylime>



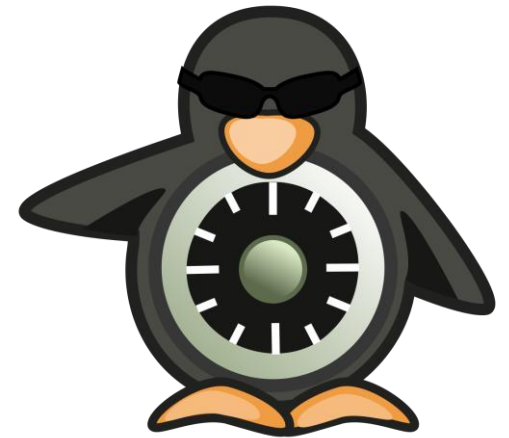
SELinux

Azzam Alaseed



Agenda

- Overview
- Architecture
- Supported AC models
- Contexts, Subjects, and Objects
- SELinux Policies
- Application support
- SELinux for Android
- Alternatives





Overview

- Brief History
 - Originally started as an extension on top of the Mach kernel by the University of Utah and the US Department of Defense as separate project.
 - The NSA later enhanced the project and made the switch to Linux.
 - Merged into mainline Linux kernel in 2001.
- What is SELinux?
 - “Security-Enhanced Linux (SELinux) is a security architecture for Linux Systems that allows administrators to have more control over who can access the system.” - RedHat
 - “A mandatory access control mechanism in the Linux kernel that checks for allowed operations after standard discretionary access controls are checked. It can enforce rules on files and processes in a Linux system, and on the actions they perform, based on defined policies” - NSA

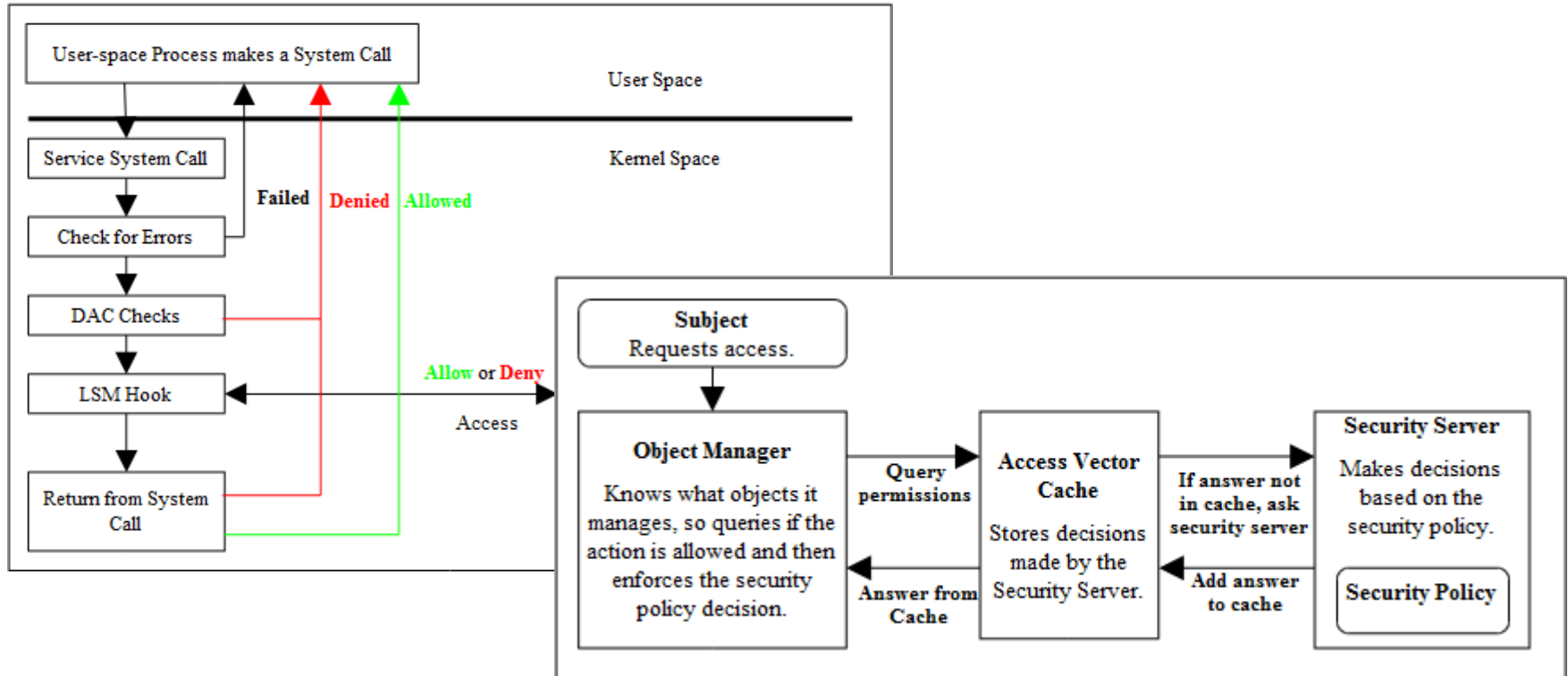


Overview

- What does it do?
 - SELinux attempts to provide information separation at the OS level.
 - Enforcement of this separation is based on the confidentiality and integrity requirements provided by the SELinux policy.
 - Help prevent information tampering and application security mechanisms bypassing.
 - Provide a confinement architecture that can limit the damage to a number of resources.
 - Incorporate a strong and flexible MAC architecture.
 - More importantly makes `chmod 777 secret_file` less damaging.



Architecture (High Level)



Source: <https://github.com/SELinuxProject/selinux-notebook>



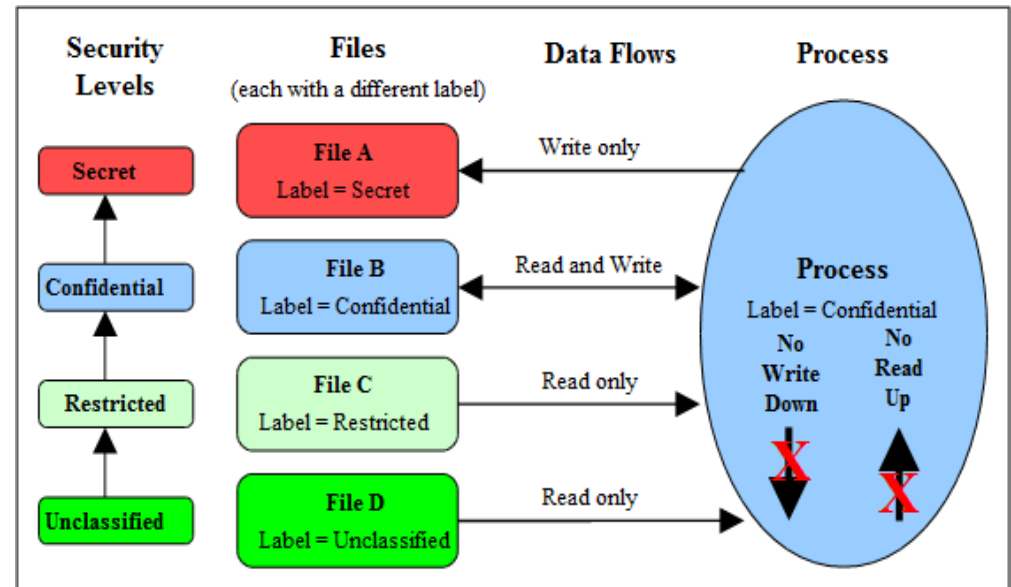
Architecture

- SELinux implements a Flask architecture
 - Flask stands for Flux Advanced Security Kernel.
 - In simple terms: it is an OS security architecture that provides flexible support for security policies.
- SELinux Components
 - Subjects.
 - Object Manager.
 - Security Server.
 - Security Policy.
 - Access Vector Cache (AVC).



Supported AC Models

- Mandatory Access Control (MAC)
 - o SELinux lets you deploy a fully functional Bell–LaPadula AC Model when running in MLS.
 - MLS has 16 levels.
 - o SELinux provides a way to get isolation by using MCS.
 - MCS has 1024 levels.

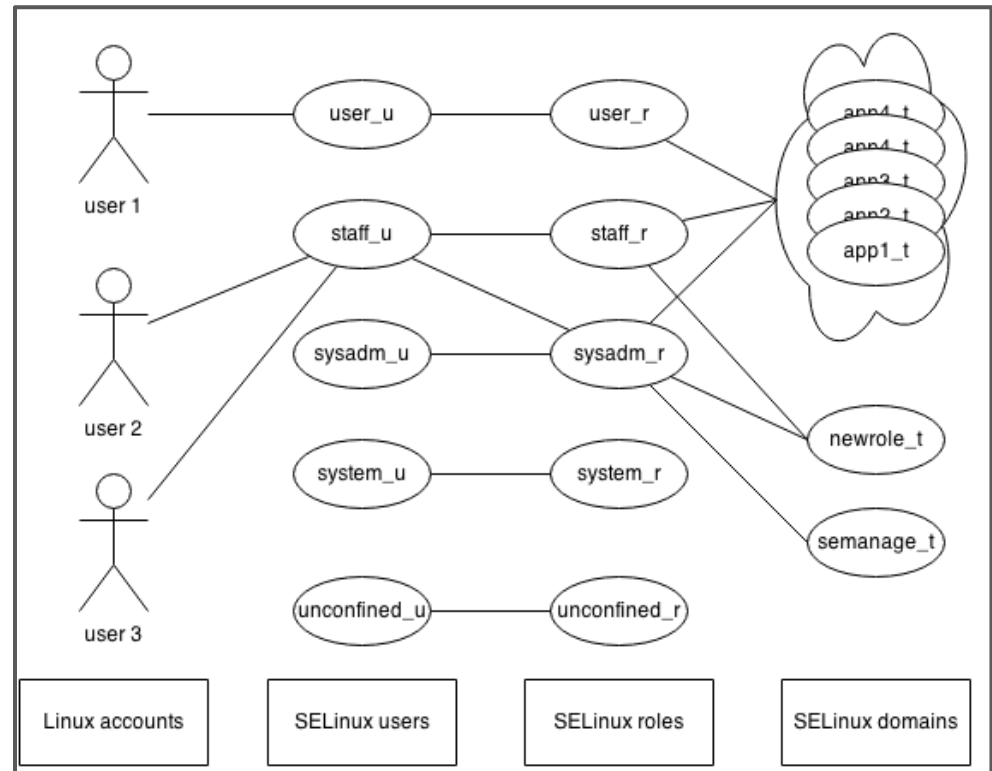


Source: <https://github.com/SELinuxProject/selinux-notebook>



Supported AC Models

- Role-Based Access Control (RBAC)
 - o SELinux lets you deploy a role-based AC policy using Type Enforcement.
 - o All subjects and objects have a type identifier associated to them.
 - o Types are basically protection domains.





Contexts, Subjects, and Objects

- Contexts

- Security Context is the glue that holds it together.
- Security Contexts comprise four fields:
 - User.
 - Role.
 - Type.
 - Range.

```

azzam@azzam-VM:~$ ls -laZ /
total 84
drwxr-xr-x. 20 root root system_u:object_r:root_t:s0 4096 Apr 12 02:00 .
drwxr-xr-x. 20 root root system_u:object_r:root_t:s0 4096 Apr 12 02:00 ..
lrwxrwxrwx. 1 root root system_u:object_r:bin_t:s0    7 Apr 12 01:33 bin -> usr/bin
drwxr-xr-x. 4 root root system_u:object_r:boot_t:s0   4096 Apr 12 01:55 boot
drwxr-xr-x. 2 root root system_u:object_r:default_t:s0 4096 Apr 12 01:35 cdrom
drwxr-xr-x. 20 root root system_u:object_r:device_t:s0 4080 Apr 12 14:51 dev
drwxr-xr-x. 131 root root system_u:object_r:etc_t:s0   12288 Apr 12 01:55 etc
drwxr-xr-x. 3 root root system_u:object_r:home_root_t:s0 4096 Apr 12 01:37 home
lrwxrwxrwx. 1 root root system_u:object_r:lib_t:s0    7 Apr 12 01:33 lib -> usr/lib
lrwxrwxrwx. 1 root root system_u:object_r:lib_t:s0    9 Apr 12 01:33 lib32 -> usr/lib32
lrwxrwxrwx. 1 root root system_u:object_r:lib_t:s0    9 Apr 12 01:33 lib64 -> usr/lib64
lrwxrwxrwx. 1 root root system_u:object_r:lib_t:s0   10 Apr 12 01:33 libx32 -> usr/libx32
drwx-----. 2 root root system_u:object_r:lost_found_t:s0 16384 Apr 12 01:33 lost+found
drwxr-xr-x. 3 root root system_u:object_r:mnt_t:s0    4096 Apr 12 01:53 media
drwxr-xr-x. 2 root root system_u:object_r:mnt_t:s0    4096 Feb 9 10:47 mnt
drwxr-xr-x. 3 root root system_u:object_r:usr_t:s0    4096 Apr 12 01:53 opt
dr-xr-xr-x. 247 root root system_u:object_r:proc_t:s0  0 Apr 12 02:01 proc
drwx-----. 4 root root system_u:object_r:user_home_dir_t:s0 4096 Apr 12 01:49 root
drwxr-xr-x. 33 root root system_u:object_r:var_run_t:s0 940 Apr 12 14:44 run
lrwxrwxrwx. 1 root root system_u:object_r:bin_t:s0    8 Apr 12 01:33 sbin -> usr/sbin
drwxr-xr-x. 8 root root system_u:object_r:default_t:s0 4096 Feb 9 10:57 snap
drwxr-xr-x. 2 root root system_u:object_r:var_t:s0    4096 Feb 9 10:47 srv
dr-xr-xr-x. 13 root root system_u:object_r:sysfs_t:s0  0 Apr 12 02:01 sys
drwxrwxrwt. 17 root root system_u:object_r:tmp_t:s0   4096 Apr 12 18:43 tmp
drwxr-xr-x. 14 root root system_u:object_r:usr_t:s0   4096 Feb 9 10:48 usr
drwxr-xr-x. 14 root root system_u:object_r:var_t:s0   4096 Feb 9 10:56 var
    
```



Contexts, Subjects, and Objects

- Subjects

- In SELinux a subject is an active entity that can cause information to flow between objects.
- Subjects in SELinux have a security context associated with them.
- Subjects can transition between domains when allowed by policy..
- Kinds of subjects in SELinux:
 - Trusted.
 - Untrusted.

LABEL	PID	TTY	STAT	TIME	COMMAND
system_u:system_r:init_t:s0	1	?	Ss	0:02	/sbin/init splash
system_u:system_r:kernel_t:s0	2	?	S	0:00	[kthreadd]
system_u:system_r:systemd-resolved_t:s0	556	?	Ss	0:00	/lib/systemd/systemd-resolved
system_u:system_r:avahi_t:s0	572	?	Ss	0:00	avahi-daemon: running [azzam-VM.local]
system_u:system_r:systemd-busd_t:s0	578	?	Ssl	0:02	/usr/bin/dbus-daemon --system --addr
system_u:system_r:NetworkManager_t:s0	580	?	Ssl	0:01	/usr/sbin/NetworkManager --no-daemon
system_u:system_r:syslogd_t:s0	606	?	Ssl	0:00	/usr/sbin/rsyslogd -n -iNONE
system_u:system_r:systemd-logind_t:s0	612	?	Ss	0:00	/lib/systemd/systemd-logind
system_u:system_r:xdm_t:s0	920	?	Ssl	0:00	/usr/sbin/gdm3
system_u:system_r:xdm_t:s0	1451	?	Sl	0:00	gdm-session-worker [pam/gdm-password]
system_u:system_r:init_t:s0	1607	?	Ss	0:00	/lib/systemd/systemd --user
unconfined_u:unconfined_r:unconfined_t:s0	1682	?	Sl	0:00	/usr/bin/gnome-keyring-daemon
system_u:system_r:initrc_t:s0	1687	?	Ssl	0:00	/usr/libexec/gvfsd
system_u:system_r:init_t:s0	2104	?	Ssl	0:23	/usr/bin/gnome-shell
system_u:system_r:initrc_t:s0	2581	?	Rsl	0:05	/usr/libexec/gnome-terminal-server
system_u:system_r:initrc_t:s0	2589	pts/0	Ss	0:00	bash
unconfined_u:unconfined_r:xserver_t:s0	1752	tty2	00:00:21	Xorg	
unconfined_u:unconfined_r:unconfined_t:s0	1878	tty2	00:00:00	gnome-session-b	
system_u:system_r:initrc_t:s0	3721	pts/0	R+	0:00	ps -axZ



Contexts, Subjects, and Objects

- Objects

- In SELinux an object is a resource that is accessed by subjects.
- Objects in SELinux consists of class identifier and an access vector.
- Objects can be labeled with the desired type (domain).
- Moved objects maintain their type while copied objects take the target type.
- Objects can transition between domains.

```

azzam@azzam-VM:~$ ls -laZ /etc/ssh/
total 584
drwxr-xr-x.  4 root root system_u:object_r:etc_t:s0 4096 Apr 12 19:55 .
drwxr-xr-x. 131 root root system_u:object_r:etc_t:s0 12288 Apr 12 19:55 ..
-rw-r--r--.  1 root root system_u:object_r:etc_t:s0 535195 Mar  9 06:17 moduli
-rw-r--r--.  1 root root system_u:object_r:etc_t:s0 1603 May 29 2020 ssh_config
drwxr-xr-x.  2 root root system_u:object_r:etc_t:s0 4096 May 29 2020 ssh_config.d
-rw-r--r--.  1 root root system_u:object_r:etc_t:s0 3289 Mar  9 06:17 sshd_config
drwxr-xr-x.  2 root root system_u:object_r:etc_t:s0 4096 Mar  9 06:17 sshd_config.d
-rw-.....  1 root root system_u:object_r:ssh_key_t:s0 505 Apr 12 19:55 ssh_host_ecdsa_key
-rw-r--r--.  1 root root system_u:object_r:etc_t:s0 175 Apr 12 19:55 ssh_host_ecdsa_key.pub
-rw-.....  1 root root system_u:object_r:ssh_key_t:s0 399 Apr 12 19:55 ssh_host_ed25519_key
-rw-r--r--.  1 root root system_u:object_r:etc_t:s0 95 Apr 12 19:55 ssh_host_ed25519_key.pub
-rw-.....  1 root root system_u:object_r:ssh_key_t:s0 2602 Apr 12 19:55 ssh_host_rsa_key
-rw-r--r--.  1 root root system_u:object_r:etc_t:s0 567 Apr 12 19:55 ssh_host_rsa_key.pub
-rw-r--r--.  1 root root system_u:object_r:etc_t:s0 342 Apr 12 19:55 ssh_import_id
  
```



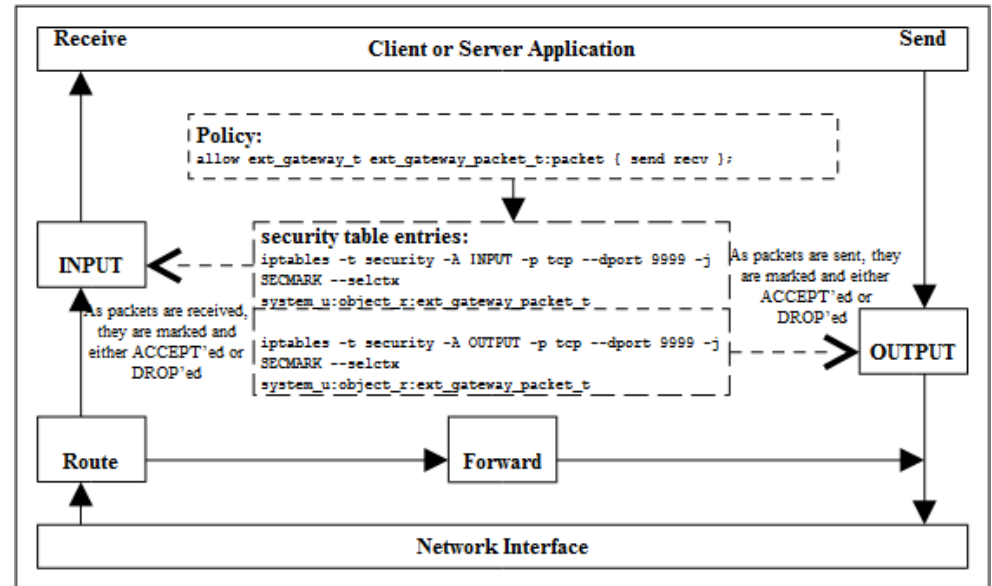

SELinux Policies

- Policies in terms of functionality
 - Minimum: small set of daemons are confined within their domains.
 - Targeted: more daemons, areas, and users are confined.
 - MLS: enable support for multi-level security with labels and clearances.
- What if you want more?
 - SELinux has tools for:
 - Standard policies have switches allowing you to turn some features on or off.
 - semanage lets add more rule to your currently active policy.
 - Build a policy from the source that best fits your needs.



Application Support

- Networking Support
 - SELinux has two hooks for controlling network access:
 - Socket level
 - The netif class to restrict interfaces.
 - The node class to restrict IP addresses.
 - The <proto>_socket classes to restrict protocols.
 - Packet processing level
 - SECMARK extension.
 - IPv4 CIPSO / IPv6 CALIPSO.
 - Labeled IPsec.

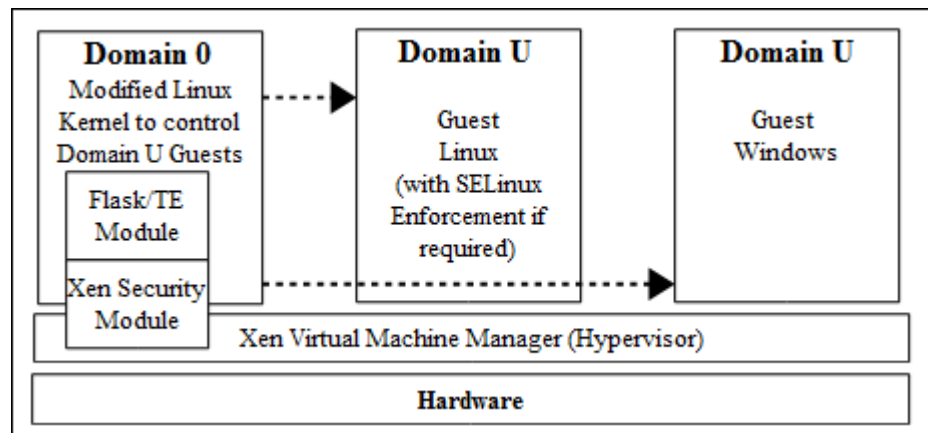


Source: <https://github.com/SELinuxProject/selinux-notebook>



Application Support

- Virtual Machine Support
 - SELinux is supported by Linux KVM and QEMU.
 - Xen has its own security module (XSM):
 - Prevent two domains from communicating.
 - Grant a set of privileges to unprivileged domains.
 - Control domain's ability to use device passthrough.
 - Restrict or audit operations.
 - Prevent privileged domains from mapping memory pages.
 - Isolate the hypervisor components.



Source: <https://github.com/SELinuxProject/selinux-notebook>



Application Support

- X-Windows Support
 - Fine-grained access control to X-Server objects.
- PostgreSQL Support
 - Adds MAC to database objects.
- Apache Support
 - Allows a web application (child process) to run with smaller set of privileges than Apache (the parent) using typebounds.

database context = 'unconfined_u:object_r:postgresql_db_t:s0' This context is inherited from the database directory label - ls -Z /var/lib/pgsql/data		
schema (db_schema) security_label = 'unconfined_u:object_r:sepgsql_schema_t:s10'		
table (db_table) security_label = 'unconfined_u:object_r:sepgsql_table_t:s0:c20'		
	column 1 (db_column) security_label = 'unconfined_u:object_r:sepgsql_table_t:s0:c30'	column 2 (db_column) security_label = 'unconfined_u:object_r:sepgsql_table_t:s0:c40'

Source: <https://github.com/SELinuxProject/selinux-notebook>



SE for Android

- Hooks in the various Android classes.
 - Builds in top of SELinux.
 - Per-file labeling.
 - Flexible labeling of Apps.
 - Java JNI Bindings to provide SELinux functionality to Android Apps.
 - Confinement of system services.



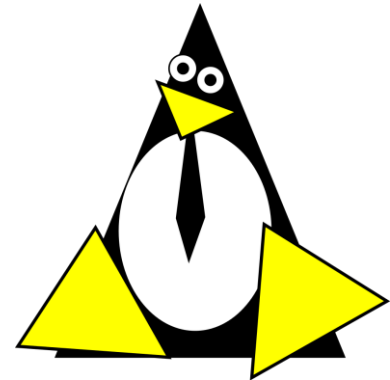


Alternatives

- AppArmor.
 - Uses profiles to specify access control for processes.
 - Works on any filesystem.
 - Enforcement is based on file path and not the inode.



- SMACK.
 - Less granular access control.
 - Suitable for embedded/IoT devices.





References (SELinux)

- <https://github.com/SELinuxProject>
- https://freecomputerbooks.com/books/The_SELinux_Notebook-4th_Edition.pdf
- <https://www.sans.org/reading-room/whitepapers/linux/introduction-nsas-security-enhanced-linux-selinux-232>
- <https://www.cs.utah.edu/flux/fluke/html/flask.html>
- http://www.selinuxproject.org/page/Main_Page
- <https://www.starlab.io/blog/a-brief-tour-of-linux-security-modules>
- http://www.asprom.com/technologie/mentorpaper_4.pdf
- https://wiki.gentoo.org/wiki/SELinux/Role-based_access_control
- <https://www.electronicsworld.com/blogs/eyes-on-android/what-is/se-android-2013-03/>
- https://wiki.xenproject.org/wiki/Xen_Security_Modules:_XSM-FLASK

DSci526: Secure Systems Administration

Accreditation and Acceptance Testing

Prof. Clifford Neuman

Lecture 12
14 April 2021
Online



DSci526: Secure Systems Administration

Accreditation and Acceptance Testing

Prof. Clifford Neuman

Lecture 12
14 April 2021
Online

Configuration Management Change Management



- Acceptance Testing is a component of change management, which is itself part of configuration management.
 - Deciding whether it is acceptable to run a newly acquired software or hardware system.
 - It should include a security component, although more often focused on functional requirements.



Acceptance Based on

- **Certification**
 - Evaluation and testing to demonstrate the system demonstrates certain properties or meets various development or testing metrics.
- **Accreditation**
 - An assessment that the system is suitable to access specific data, and in a particular environment.
- **Acceptance Testing**
 - Testing in the context of your system to demonstrate conformance to functional and security criteria. This is usually a final step before payment is made and the system is granted authority to operate on live data.



Acceptance Testing

- Applied to systems purchased (usually as custom developed by another organization).
 - Why not develop in house:
 - Cost
 - Cheaper to buy components and systems than to design and build from scratch.
 - Convenience/ Expertise
 - Creator might recognize it is easier to buy a component or system already on the market than design one
 - Creator might recognize they do not have the expertise to design the needed system
 - You must still develop or acquire the expertise to evaluate the security of the system.

Trust of your systems (scenarios)



- Trust by own design
- Trust by using trusted supplier
- Trust by testing



Trust By Own Design

- Least risk that something malicious will be introduced to the system.
 - Often greater risk that security flaws remain.
 - Trust depends on the confidence in design/development.
- You will likely end up relying at least on hardware obtained elsewhere, unless you have significant capabilities to build your own chips.

Trust By Using Trusted Supplier: Accreditation



- Obtain components, subsystems or entire systems from a proven, trusted supplier.
 - Approved for access to your data, and your environment.
 - Supplier has proven over time to provide functional, secure products (maturity)
 - Trust can be gained through reputation of product or supplier
- Concern: Supply Chain Attacks

Trust By Testing: Acceptance Testing



- For component, subsystem or system not trusted either by design or coming from a trusted supplier.
 - Or as additional testing before “going live”
 - Traditionally focused on Functionality Testing
 - Should include Security Testing

Authorization Process within Air Force

- Other branches of military use similar processes, also similar processes for functionality. Will focus on Cyber Security process
- Goal: Obtain ATO (Authorized to Operate) or ATC (Authorized to Connect)
 - ATO typically for Aircraft, ATC typically for support equipment
- Process for Cyber Security: Risk management Framework (RMF)

Risk Management Framework



- 1. Categorize System
- 2. Select Security Controls
- 3. Implement Security Controls
- 4. Assess Security Controls
- 5. Authorize Information Systems
- 6. Monitor Security Controls



1. Categorize System

- A categorization package is created that describes the system
- System Architecture, Data Flows and an initial risk assessment (mainly documenting possible threats to the system) are created.
- There are different authorizing officials (AO) and technical advisors for different types of systems



2. Select Security Controls

- Security Controls are Security Mechanisms
 - Technical Controls- Ex. Lock out user after 3 incorrect logins
 - Operational Controls- Ex. System not to be used for unauthorized purposes
 - Management Controls- Ex. Incident Response
- Baseline List of Security Controls created by NIST
- Each directorate (system type) has a general guideline to what security controls are applicable to that type of system
- Program will use these guidelines to create a list of the security controls applicable to their system

3. Implement Security Controls



- Take the controls chosen to be applicable to the system and implement them, if not already already implemented
- Update documents for system architecture, software and hardware lists to include new security controls



4. Assess Security Controls

- Test (and document) results of security controls
- This mainly is used for technical security controls
- Risk assessment is now updated with vulnerabilities (security controls that have not been implemented). Vulnerabilities that have a threat matched to them produce a risk.
- Risk is evaluated and proven to show it is at an acceptable level (or there are actions that can be taken to lower the risk to an acceptable level)



4. Assess Security Controls

- This is where accreditation & acceptance testing comes in
- For most companies equipment is bought, not built. Testing is vital to ensure equipment behaves as needed.
- If the system plans to use a commonly used piece of equipment generally less testing is needed (system or component may already be accredited for environment).
- However, if a program is planning on using a unique piece of equipment, more testing is needed (acceptance testing)

4. Assess Security Controls



- Any unique equipment needs more analysis to decide if it is secure. Acceptance is not just based on the equipment itself, but also the maintenance and other factors

5. Authorize Information Systems



- Risk Assessment and POAM (plan of action and milestones) are presented to an Authorizing Official
- Authorizing Official either decides risk is acceptable (and assumes the risk) and gives program an ATO, or decides risk is unacceptable and more security needs to be implemented to the system.
- Authorizing Official is familiar if equipment is accredited or not, can tell program office more acceptance testing is needed.



6. Monitor Security Controls

- Once system is given ATO, program must now monitor that the security controls are being implemented properly
- ATO's must be renewed (typically one or two years)
- Program continuously goes through RMF process (skipping step 1), sometimes having to select new updated security controls



Authorization Process: Accreditation

- Accreditation works in two ways within the authorization process
- 1. Accreditation of components or subsystems being bought requires less acceptance testing.
- 2. ATO is an accreditation. Once system receives ATO, it is accredited that all of the organization will recognize this system's ability to operate securely for a defined environment.

Accreditation and Acceptance Testing in Industry



- Industries also must perform some sort of testing on products they buy
- However, industries typically put more emphasis on functionality and availability than security (Microsoft acceptance testing example)
- Accreditation in industry is related to who a company will purchase from
- Acceptance Testing in industry used more as a way to validate a contract and provide payment



Accreditation in Industry

- Industries tend to buy from established companies that have proven to provide products that work
- Example: Microsoft Office
- However, this also applies to when companies need new software built for them.



Accreditation in Industry

- Software purchase agreements are made whenever a company is purchasing software.
- Accreditation comes into play in a couple of ways.
 1. Company might only be willing to buy software from an accredited source
 2. Company might give me leeway on a contract given to an accredited source (in how much acceptance testing is needed before

Acceptance Testing in Industry



- Companies often provide contracts to a different company to build something they need.
- Acceptance Testing is used to:
 - 1. Keep the company on contract on track
 - 2. Provide a concrete way to test the product, if it does not pass the tests the company won't get paid the full amount

Acceptance Testing in Industry



- For a big system, acceptance testing is very important
- A company will provide in the contract tests it will perform on the product before the final purchase is made.
- Typically functionality related, but security is becoming a bigger part.

Microsoft Acceptance Testing



Acceptance Testing

A common form of acceptance testing consists of tests that exercise a user scenario from the user interface. These tests emulate the application keyboard and user interface interactions. A user interface test for a browser-based application is known as a Web test. Microsoft Visual Studio Team System includes automation for navigation that allows you create user interface tests. The Partner Portal application uses the following tests to conduct three forms of acceptance testing:

- **Build verification testing.** This is a regular check of application functionality from the perspective of an end user scenario of the application under development. For a list of build verification tests used by the Partner Portal application, see [Build Verification Tests in the Partner Portal Application](#).
- **Stress testing.** This is a test of the application to handle a very high volume of user scenarios being executed concurrently. The purpose of stress testing is to drive the system beyond expected maximum load in production to see how it behaves. For more information, see [Stress Testing](#).
- **Scale testing.** This is a test of the application to determine whether the application can handle the required number of users under normal load conditions on the production hardware configuration. For more information, see [Scale Testing](#).

For more information about automated tests, see [How to: Create Automated UI Test Methods for SharePoint](#). For more information about using Visual Studio for testing Web applications, see [Introducing Microsoft Visual Studio 2005 Team System Web Testing](#).



Apple Acceptance Testing

- Looking for user acceptance testing
- This implies functionality (a long with usability)
- Security acceptance testing is also done, and a list of certifications and validations can be found online (link provided shows acceptance testing and certificates for cryptographic methods used in iOS).
 - <https://support.apple.com/en-us/HT202739>

Accreditation and Acceptance Testing

- Accreditation is used to provide trust without the need for additional (possibly costly) testing
- Acceptance Testing provides trust that is absent when no accreditation is in place.



DSci526: Secure Systems Administration

Case Studies in Administration

Prof. Clifford Neuman

Lecture 12
14 April 2021
Online

Assigned Reading For 21 April



- Please skim and read the introduction and relevant sections (the ones labeled FY 2018 Inspector General FISMA Report) from:
 - [FEDERAL CYBERSECURITY: AMERICA'S DATA AT RISK STAFF REPORT PERMANENT SUBCOMMITTEE ON INVESTIGATIONS UNITED STATES SENATE](#)
 - We will use this as the basis of discussion of case studies of unsecure system administration.
 - Please think about which aspects of secure system administration as covered in this class were not properly applied, and what should be done instead.



For Discussion Now

Report shows failures at eight US agencies in following cyber-security protocols

- US Senate report finds appallingly bad cyber-security practices at eight US government agencies.
 - ZDNet – June 26 2019
 - Catalin Cimpanu for Zero Day



DSci526: Secure Systems Administration

Second Group Project
Third Week Discussion

Prof. Clifford Neuman

Lecture 12
14 April 2021
Online



Wrapping up Project Two

- It is time to wrap up exercise Two. By Monday 26 April - each group should prepare a report describing:
 - User documentation for their application (high level)
 - Their network and server architecture (what servers are on what VM's and how they are interconnected)
 - A risk assessment/vulnerability analysis enumerating the risks, explaining the mitigation of those risks, and listing those threats that are not defended against (i.e. where you accept the risks).
 - A description of the steps taken for pen testing of your system.
 - On Wednesday 28 April, your team will have 25 minutes to present this summary to the entire class (this time, no withholding of information from the other team)
 - This week and next, basic 5 minute presentation, and Break Out Groups.
- We will use time in the final lecture to demonstrate the operation of your systems.
 - Please prepare a list of tests (with appropriate scripts) that you believe should be run against your system, and the other team's system, and send me that list of tests by Monday 26 April.

Teams for Second Group Project



- Team One

- Shagun Bhatia
- Anthony Cassar
- Sarahzin Chowdhury
- Tejas Kumar Pandey
- Pratyush Prakhar
- Christopher Samayoa
- Louis Uuh
- Ayush Ambastha
- Jason Ghetian
- Abhishek Tatti
- MaryLiza Walker
- Hanzhou Zhang

- Team Two

- Azzam Alsaeed
- Marco Gomez
- Alejandro Najera
- Doug Platt
- Carol Varkey
- Yang Xue
- Aditya Goindi
- Malavika Prabhakar
- Dwayne Robinson
- Amarbir Singh
- Shanice Williams

Second Exercise - Criminal Enterprises

- Chosen because of differences in the high-level principles.
 - Not because I expect you to implement these kinds of systems in your future endeavors.
 - But you may be called upon to break some of these systems if later employed by government organizations.
- Your organization must:
 - Accept Bitcoin as payment (not really, but it must accept something that stands in for bitcoin)
 - Manage an inventory of stolen account identifiers with passwords
 - Enable the sale of collection of such information in exchange for your stand-in for bitcoin
 - Control access to such information
 - Prevent collection of evidence or intelligence by third parties.
 - Note, do not deal in any illegal goods, but use dummy information to stand in for such goods. Also, do not use terms associated with such illegal goods or information in communications, make up new names for this dummy information.