VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

GENERAL NOTIFICATION SYSTEM FOR FREEIPA

SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE Bc. PETR KUBÁT
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

OBECNÝ NOTIFIKAČNÍ SYSTÉM PRO PROJEKT FREEIPA
GENERAL NOTIFICATION SYSTEM FOR FREEIPA

SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE AUTHOR	Bc. PETR KUBÁT
VEDOUCÍ PRÁCE SUPERVISOR	Mgr. ADAM ROGALEWICZ, Ph.D.

BRNO 2015

Abstrakt
Výtah (abstrakt) práce v českém jazyce.

Abstract
Výtah (abstrakt) práce v anglickém jazyce.

Klíčová slova
LDAP, Active Directory, FreeIPA, Kerberos, DNS, Dogtag

Keywords
LDAP, Active Directory, FreeIPA, Kerberos, DNS, Dogtag

Citace
Petr Kubát: General Notification System for FreeIPA, semestrální projekt, Brno, FIT VUT v Brně, 2015

General Notification System for FreeIPA

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Mgr. Adama Rogalewicze, Ph.D.

....
Petr Kubát
January 7, 2016

Poděkování

Rád bych poděkoval hlavně panu Petru Špačkovi za jeho trpělivost pri odborném vedení práce.

Contents

Chapter 1

Introduction

Chapter 2

FreeIPA

FreeIPA (where IPA stands for Identity, Policy and Audit) is an open-source security management solution sponsored by Red Hat aimed primarily at Linux and Unix machines[11].

The project itself combines a number of various existing open-source technologies to achieve the goal of providing centralized authentication and authorization, as well as storing important account information like users or group memberships. FreeIPA also aims to provide easy management and setup of a domain controller which would otherwise be very difficult by using the same components on your own.

In this chapter I will briefly introduce some of the components FreeIPA uses and describe the architecture of the resulting FreeIPA server solution.

2.1 Directory Server

FreeIPA's directory service is the foundation of the project as it stores various information on behalf of all of FreeIPA's components. It also plays a big role in authentication and authorization using Kerberos which will be presented in the next section.

The LDAP protocol[10] is used as a means of communication with the server and the data itself is stored in a Directory Information Tree (DIT) which is a tree-like data structure. An example of a DIT structure can be seen in figure 2.1.

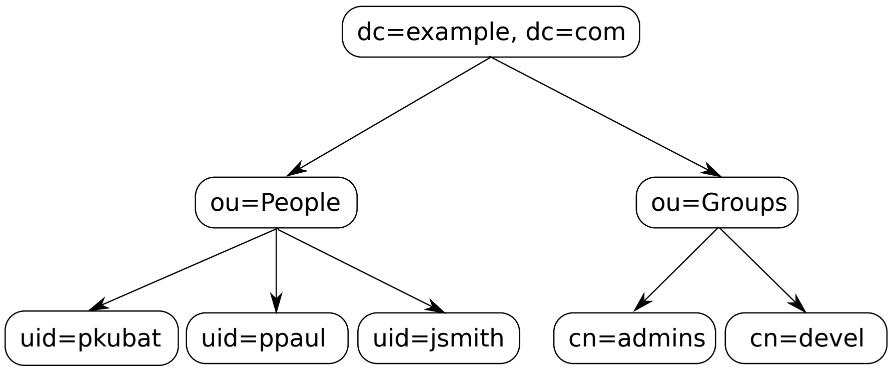

Figure 2.1: LDAP Directory Information Tree

LDAP provides several operations to use with the server[10]:

- **add, delete, modify:** These operations add, remove and modify the data contained in the DIT.

- **search, compare:** The search and compare operations are used in querying the DIT for specific information.
- **bind, unbind, abandon:** These operations can be used to authenticate to the directory, terminating the connection or abandoning a previously sent request entirely, respectively.
- **extended operations:** New operations that are not a part of the original protocol.

The actual LDAP compatible server contained in FreeIPA is implemented using the 389 Directory Server project[3].

2.2 Kerberos

Kerberos[9] is a network authentication protocol that uses symmetric encryption using a pre-shared key to authenticate the client to a network service (and vice versa) via an insecure connection using a trusted third party service called a Key Distribution Center (KDC). The resulting communication is secure because no secret keys are transported over the network in plaintext format as the KDC already contains a database of credentials for users and services in the Kerberos realm.

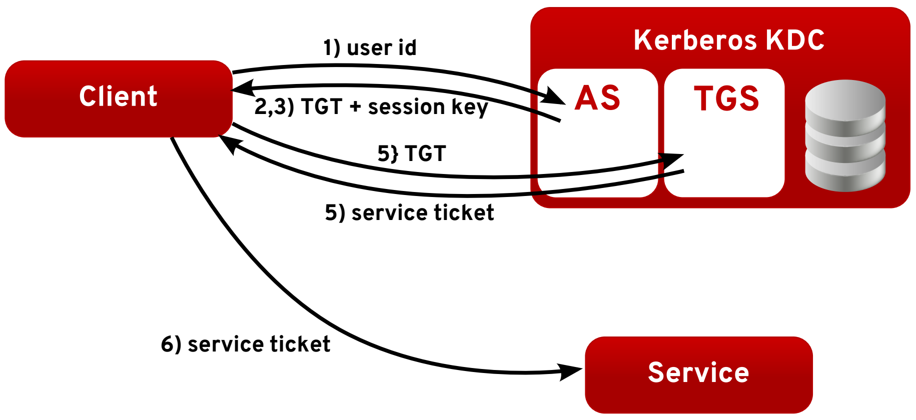

Figure 2.2: Kerberos authentication process

The process of authenticating the user to a network service is shown in figure 2.2 and can be described in these steps:

1. The user sends his principal name (an unique identifier) to the KDC's Authentication Server (AS) via a plaintext request.
2. The AS then checks the database to make sure the user exists and sends back a randomly generated session key to be used to encrypt communication with another service called a Ticket-Granting Service (TGS) encrypted with the user's secret key.
3. The AS also generates a set of credentials called a Ticket-Granting Ticket (TGT) which includes the previously generated session key and is encrypted by the secret key of the TGS.

4. After recieving the first message the client decrypts it using his secret key. This is the only time the user's key is actually used. The TGT which the client can't decrypt himself is saved in a cache on the client's side to be used later to setup a session with the TGS. At this point the user is authenticated to the Kerberos realm and doesn't have to input his secret key again for a set amount of time (commonly 10-24 hours).
5. When the user wants to authenticate agains a service in the Kerberos realm he just has to ask the TGS to send him a ticket.
6. The user then authenticates to the chosen service using this ticket without the need for his secret key.

As the security of the Kerberos protocol is partly based on the time stamps of tickets, all of the clients and services in the realm have to be properly synchronized time-wise. To achieve this goal the Network Time Protocol can be used in the FreeIPA project.
FreeIPA's KDC is implemented using the MIT Kerberos[7] open source software and FreeIPA also provides its own KDC data backend called ipa-kdb which is used to both read and write user information to FreeIPA's LDAP directory service[12].

2.3 DNS

Even though it would be possible to access network services located in a FreeIPA domain directly using their IP addresses, it is much more easier to do so using domain names.
The Domain Name System (DNS)[8] is distributed naming system, that translates domain names, which can be easily memorized by humans, into IP addresses using special name servers. As such if one wants to access a network service or a webpage he doesn't have to remember its IP address, only the IP address of the name server (which is stored localy on the client machine) and the domain name of the service/webpage.
The domain name space resembles a tree structure, each node having a label that designates a part of its domain name, while the full domain name of the node can be built by concatenating this label with the domain name of its parent node.
The name space is divided into zones starting at the root of the tree structure with child nodes of the root node called top-level domains (TLD).

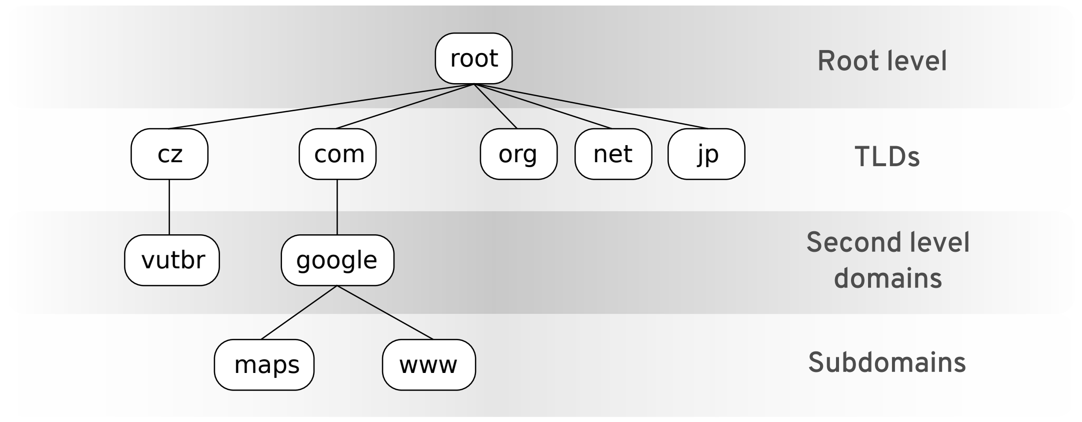

Figure 2.3: Domain name space

These zones can contain one or several domains, each domain served by one or several name servers, and can be divided into additional zones if deemed necessary.
The DNS server in FreeIPA uses an enhanced BIND name server which allows FreeIPA to store data into an LDAP directory[13]. However using FreeIPA's integrated DNS server is optional and as such the project can be used with a different third party DNS server if so desired.

2.4 FreeIPA Architecture

While the most important components of the FreeIPA project have been described in previous sections, a number of components are still missing for the infrastructure to be complete.

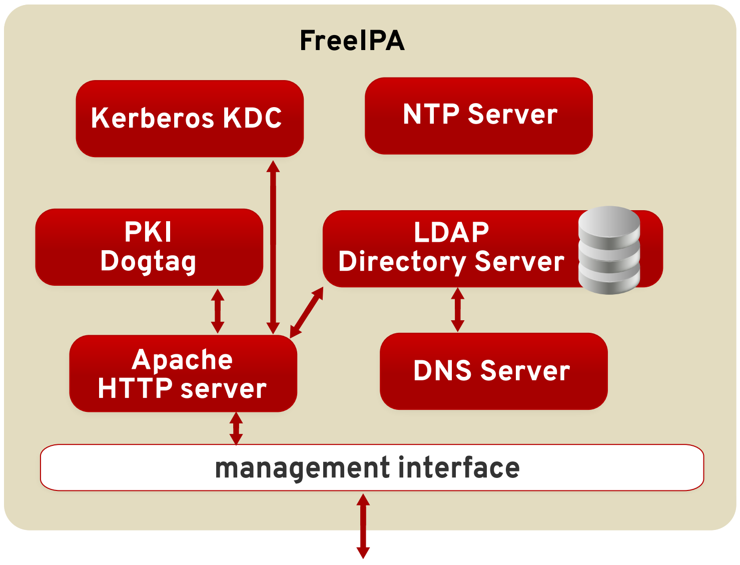

Figure 2.4: High level FreeIPA server architecture

To start with, FreeIPA can optionally use the Dogtag project[2] as an integrated Public Key Infrastructure that signs and publishes certificates for hosts and services inside the domain.
Secondly, the Apache Web Server is used to provide web based access to the management APIs using the XML-RPC and JSON-RPC APIs as well as to serve the web interface to the clients.
And last but not least, as one of the most important additions to the project, FreeIPA's ipalib framework is used together with the Command Line and Web based interfaces to administer the FreeIPA domain.
A high level and a detailed representation of the resulting infrastructure can be found in figures 2.4 and 2.5, respectively.
The ipalib framework is written using the Python programming language and is highly modular – most if not all of its functionality is implemented using plug-ins, which are executed on demand using the Apache Server. This framework is generally used to connect the isolated components into the FreeIPA solution, providing the ability to enroll users or services into the domain, managing the certificates used within the domain or adding Host Based Access Control rules for even beter access management.

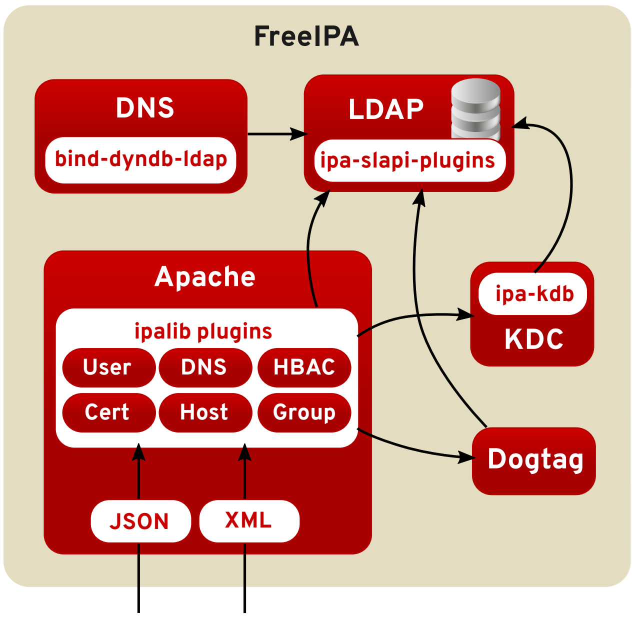

Figure 2.5: Detailed look at FreeIPA architecture

2.5 Extending FreeIPA

2.5.1 Extending the Framework

As previously stated, the ipalib framework is highly modular and extensible with a couple of different ways to add new (or modify current) functionality[1]:
First off one can directly extend existing objects of the ipalib framework. These objects are mainly responsible for executing various commands sent using the XML and JSON APIs and as such extending or modifying the objects, eg. adding a parameter or rewriting an object's label, can alter the default behaviour of those commands.
The second way is to extend methods of an object stored in the LDAP database by adding callbacks to these methods. Callbacks are user-defined functions that are called at various stages of exectution of the method:

Pre callback
This callback is called before executing the method's code. Used for modifying arguments and data validation.

Post callback
This callback is called after executing the method's code and allows for analyzing the result of the command.

Exc callback
This callback is called in case there is an error during the execution of the method's code. Can be used to recover from the error.

Interactive callback
Allows a command to decide if additional parameters should be requested from the user.

2.5.2 Extending the Directory Server

The 389 DS used in the FreeIPA project has only one way of creating extensions and that is using server plug-ins. There are several types of plug-ins that can be used when trying to extend the directory server[14]:

Pre operation
A pre operation plug-in is executed before starting the LDAP operation. Mostly used for data validation.

Post operation
A post operation plug-in is executed after performing the LDAP operation. Can be used for notifications.

Entry storage and fetch
Executed before writing and reading data from the database, respectively. An example of this type of plug-in would be encryption/decryption of data saved in the database.

Extended operation
Executed when the client calls an extended operation.

Syntax
Run when getting a list of candidates for a search. Can be used to modify comparison operations.

Matching rule
Run when the client sends a request with an extensible matching search filter.

One can see that there are some similarities to callbacks of ipalib's extensions, especially the way of calling pre and post operations plug-ins is basically the same as pre and post callbacks differing only in when they are called. While pre and post callbacks are bound to a specific method of a specific command object (eg. before or after executing a user-add command), the pre and post operations plug-ins are bound to an LDAP operation and as such are run much more frequently.

Chapter 3

Active Directory

3.1 Structure and Components

This section is based on the official documentation for Active Directory[6].
AD is a distributed directory service used by the Microsoft Windows Server operating systems and, while similar in the components used, is quite different to the FreeIPA project. Objects located in a AD organized network are structured into a logical structure, the basis of which form forests and domains.
Forests are the security boundaries of the logical structure and can be structured to provide data and service autonomy and isolation, removing the dependency on physical topology. Each forest is able to hold several domains. These domains can be structured to provide data and service autonomy but not isolation as to allow for replication optimalization. This logical structure can also be used to control access to data for each domain seperately.

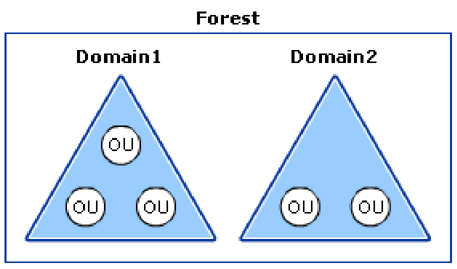

Figure 3.1: An Active Directory forest and its domains[6]

Each domain is connected to one or more domain controllers. A domain controller is a service that stores directory data and manages user and domain interactions. Each domain controller consists of a schema, which defines objects and attributes that can be store in the directory, a data store, which managaes the storing and retrieving of data, and the actual database. The data store uses a number of interfaces to access stored data as can be seen in figure 3.2, one of which is the same protocol as the 389DS uses in FreeIPA – LDAP.

Active Directory also uses, similarly to FreeIPA, DNS as its domain controller location mechanism and Kerberos for user authentication.

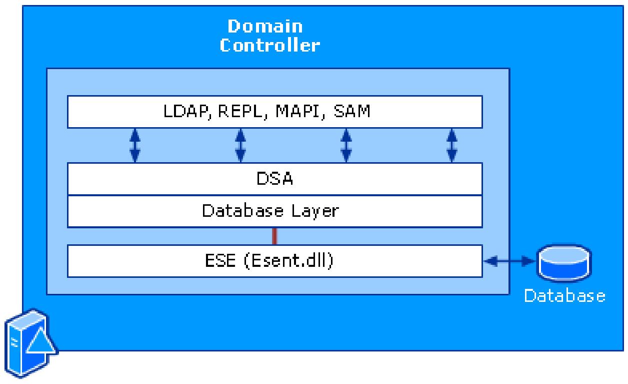

Figure 3.2: Active Directory Domain Controller[6]

3.2 Notification Mechanisms

As AD does not by itself provide a complete notification system, this section describes mechanisms contained in AD and Windows operating systems in general that can be used to send notifications between applications.
The following mechanisms are available for users of AD directory service:

Change Notifications

This mechanism allows for a client to register with a domain controller to recieve notifications for changes in the AD domain service. The notification control is specified in a LDAP asynchronou search operation.
The user can define a number of search parameters: scope, to monitor just the object or its immediate children, filter, which allows for object filtering and attributes, a list of the changed object's attributes that will be returned[5].

Polling

The second notification mechanism available in AD is directly polling the AD directory service for changes. There are two types of polling available. Polling using Directory Synchronisation control, which is an LDAP server extension that allows to check for changes since a previous state, while the other type – querying using a special uSNChanged attribute – is used when Directory Synchronisation is unavailable or inefficient[5].

These mechanisms are, however, primarily used to mantain consistency of data between the AD directory service and other applications[5]. As such there is one additional mechanism that can be used not only with AD, but generally with any application running on a Windows operating system and that is Event Tracing (ET)[4].
ET is a mechanism that allows for application defined events to be logged into a log file and optionally recieved by other applications in real time. ET has three major components, as can be seen in figure 3.3: controllers, that control the ET sessions, providers that write events into a ET session and consumers which consume one or more event sessions either directly or from a log file.

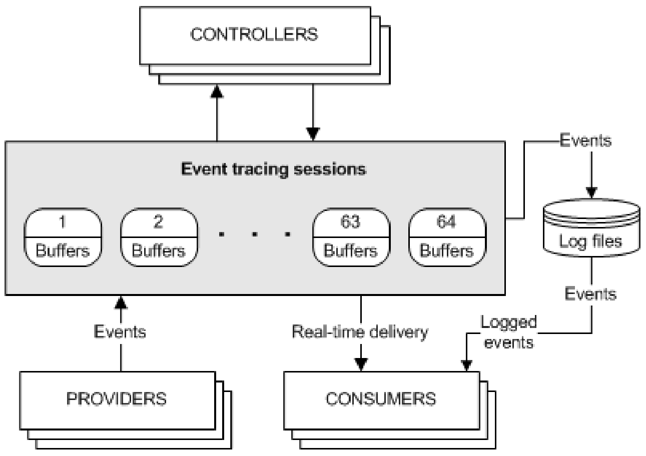

Figure 3.3: Event Tracing model[4]

Chapter 4

Analyze

Chapter 5

Conclusion

Bibliography

[1] Alexander Bokovoy. Extending FreeIPA. [online], 2011.
https://abbra.fedorapeople.org/freeipa-extensibility.pdf.

[2] Dogtag. Dogtag Certificate System. [online], [cit. 2015-12-12].
http://pki.fedoraproject.org/.

[3] Red Hat. 389 Directory Server. [online], [cit. 2015-12-12].
http://directory.fedoraproject.org/.

[4] Microsoft. Event Tracing. [online], [cit. 2015-12-12].
https://msdn.microsoft.com/en-us/library/bb968803%28v=vs.85%29.aspx.

[5] Microsoft. Tracking Changes. [online], [cit. 2015-12-12].
https://msdn.microsoft.com/en-us/library/ms677974%28v=vs.85%29.aspx.

[6] Microsoft. Active Directory Collection. [online], [cit. 2016-01-06].
https://technet.microsoft.com/en-us/library/cc780036%28v=ws.10%29.aspx.

[7] MIT. Kerberos: The Network Authentication Protocol. [online], [cit. 2015-12-12].
http://web.mit.edu/kerberos/.

[8] P. Mockapetris. Domain Names - Implementation and Speci

cation, RFC 1035. [online], November 1987 [cit. 2015-12-27].
https://tools.ietf.org/html/rfc1035.

[9] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5), RFC 4120. [online], July 2005 [cit. 2015-12-25].
https://tools.ietf.org/html/rfc4120.

[10] J. Sermersheim. Lightweight Directory Access Protocol (LDAP): The Protocol, RFC 4511. [online], June 2006 [cit. 2015-12-12]. https://tools.ietf.org/html/rfc4511.

[11] The FreeIPA Team. About FreeIPA. [online], [cit. 2015-12-12].
http://www.freeipa.org/page/About.

[12] The FreeIPA Team. Kerberos. [online], [cit. 2015-12-25].
https://www.freeipa.org/page/Kerberos.

[13] The FreeIPA Team. DNS. [online], [cit. 2015-12-27].
https://www.freeipa.org/page/DNS.

[14] Tomáš Čapek and Ella Deon Ballard. Red Hat Directory Server 10 Plug-in Guide. [online], June 2015 [cit. 2016-01-01]. https://access.redhat.com/documentation/en-US/Red_Hat_Directory_Server/10/html/Plug-in_Guide/.