



Community Experience Distilled

Open Source Identity Management Patterns and Practices Using OpenAM 10.x

An intuitive guide to learning OpenAM access management capabilities for web and application servers

Waylon Kenning

[PACKT] open source*
PUBLISHING community experience distilled

Open Source Identity Management Patterns and Practices Using OpenAM 10.x

An intuitive guide to learning OpenAM
access management capabilities for web
and application servers

Waylon Kenning



BIRMINGHAM - MUMBAI

Open Source Identity Management Patterns and Practices Using OpenAM 10.x

Copyright © 2013 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: August 2013

Production Reference: 1190813

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK..

ISBN 978-1-78216-682-5

www.packtpub.com

Cover Image by Abhishek Pandey (abhishek.pandey1210@gmail.com)

Credits

Authors

Waylon Kenning

Project Coordinator

Deenar Satam

Reviewers

Peter Major

Bino Yohannan

Proofreader

Samantha Lyon

Acquisition Editor

Vinay Argekar

Indexer

Rekha Nair

Priya Subramani

Commissioning Editor

Yogesh Dalvi

Production Coordinator

Pooja Chiplunkar

Technical Editors

Anita Nayak

Aparna Chand

Cover Work

Pooja Chiplunkar

About the Author

Waylon Kenning is an Enterprise and Solutions Architect for a large Australasian utility company with an interest in Identity Management. He currently evaluates technologies and their applicabilities within large corporate organizations.

He has worked on one of the largest Identity Management projects in New Zealand based on Sun Access Manager, which evolved into OpenAM.

I would like to thank my wife who was doubtful that I could write a book, juggle a career, and help run an ICT not-for-profit organization. You were only partially correct!

About the Reviewers

Peter Major is a true believer in open source who has been involved with OpenSSO since 2009. Since then he's been an active member of both the OpenSSO and the OpenAM community, and as from 2011 he's working at ForgeRock as a sustaining engineer for OpenAM.

Bino Yohannan has more than 6 years of experience in Identity and Access Management. He is very passionate on Web security. He has more than 10 years of experience in Information Technology. He has done his graduation in Mathematics and post graduation in Computer Applications.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit www.PacktPub.com for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<http://PacktLib.PacktPub.com>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface	1
Chapter 1: Identity Management Patterns and Principles	7
Defining Identity Management	7
How claims relate to identity	8
Understanding identity contexts	8
Why Identity Management is important?	9
Examples of identity levels	9
Pseudonymous identities	9
Trusted identities	10
Trusted identities with multiple contexts	10
Federated identities	10
How Identity Management works	10
Key components of Identity Management	12
Identity Service Providers	12
Identity policy agents	12
Identity providers	12
Identity data stores	13
Identity managers	13
Summary	13
Chapter 2: Installing OpenAM 10.x	15
Downloading OpenAM 10.x	15
Prerequisites for OpenAM	16
Creating a fully qualified domain name	16
Installing the Java Runtime Environment	17
Downloading the Tomcat application server	18
Configuring Tomcat for OpenAM	18
Installing OpenAM 10.1.0	19
Summary	25

Chapter 3: Cross-Domain Single Sign On	27
An introduction to Cross-Domain Single Sign On	27
Securing an Apache 2.4 local domain website	28
Creating an Apache Policy Agent profile in OpenAM	28
Securing Apache with the OpenAM Policy Agent	30
Securing a Tomcat 6 remote domain website	31
Configuring Tomcat and creating a Tomcat	
Policy Agent profile	31
Securing Tomcat with the OpenAM Policy Agent	33
Configuring a Tomcat Agent profile for	
Cross-Domain Single Sign On	35
Summary	36
Chapter 4: Distributed Authentication	37
Understanding distributed authentication	37
How policy agents communicate with OpenAM	37
Understanding defense-in-depth architectures	38
Preparing OpenAM for distributed authentication	38
Configuring the distributed authentication application server	41
Configuring the distributed authentication application	41
Testing distributed authentication	44
Summary	46
Chapter 5: Application Authentication with Fedlets	47
Understanding Fedlets	47
Advantages of Fedlets over policy agents	47
Disadvantages of Fedlets over policy agents	48
Configuring the Fedlet application server	48
Creating a SAML hosted identity provider	49
Creating a Fedlet	50
Deploying Fedlet.zip onto our Java application server	52
Validating the Fedlet setup	53
More information about Fedlets	55
Summary	55
Chapter 6: Implementing SAML2 Federation Patterns	57
Understanding SAML	57
Understanding Identity Providers	57
Understanding Service Providers	58
Understanding a Circle of Trust	58
Configuring OpenAM as a SAML Identity Provider	58
Installing SimpleSAMLphp	61

Configuring SimpleSAMLphp as a Service Provider	62
Configuring OpenAM to trust a SimpleSAMLphp SP	65
Testing our SAML Circle of Trust	66
Summary	67
Chapter 7: OAuth Authentication	69
Understanding OAuth	69
Preparing Facebook as an OAuth Provider	70
Configuring an OAuth authentication module	70
Configuring Authentication Chaining	75
Testing our OAuth Client against Facebook as an OAuth Provider	76
Summary	78
Chapter 8: Two Factor Authentication	79
Understanding two factor authentication	79
Understanding OATH and how it relates to OpenAM	79
Configuring OpenAM for two factor authentication	80
Configuring OpenAM to use additional LDAP attributes	80
Installing an OATH HOTP token generator	81
Populating our LDAP attributes with values	82
Configuring the OATH authentication module	83
Testing two factor authentication	85
Summary	87
Chapter 9: Adaptive Risk Authentication	89
Understanding Adaptive Risk authentication	89
Understanding how Adaptive Risk authentication works	89
Adding the Adaptive Risk module	90
Configuring the Adaptive Risk module	91
Adding adaptive risk to the authentication chain	96
Potential authentication patterns	97
Summary	97
Index	99

Preface

Identity Management is increasingly becoming one of the cornerstones of the Internet. As we interact with more and more systems, the burden of Identity Management continues to increase on users. And as the number of systems increase, the number of users increase, and the number of devices increase, and the complexity of Identity Management systems increases exponentially. This complexity of managing the authentication needs of multiple systems, federated identity repositories, and different users with different levels of risk require a centralized way of managing authentication and authorization.

Open Source Identity Management Patterns and Practices Using OpenAM 10.x shows how authentication and authorization can be managed using OpenAM, guiding you through the process of installing and configuring the application in a series of prototypes. Key concepts and technologies are covered giving you broad knowledge of the different areas of Identity Management, as well as specific examples of using Identity Management technologies such as OAuth and OATH.

Open Source Identity Management Principles and Patterns using OpenAM 10.x was written using OpenAM 10.1 using Windows 7. At the time of writing, OpenAM 10.2 is currently in testing and features specific to it are not incorporated into the book.

What this book covers

Chapter 1, Identity Management Patterns and Principles, serves as an introduction for readers new to Identity Management by covering what Identity Management is, why it is important, how it works, and what the key components of identity management are.

Chapter 2, Installing OpenAM 10.x, serves as a quick installation reference for readers new to OpenAM. This chapter covers downloading, installing, and running OpenAM for the first time.

Chapter 3, Cross-Domain Single Sign On, serves as a quick primer on what Cross-Domain Authentication is and how to achieve it with OpenAM, how it differs from Single-Domain authentication, configuring OpenAM for Cross-Domain Authentication, and cautions using the feature.

Chapter 4, Distributed Authentication, serves as a quick primer on what Distributed Authentication is and how to achieve it with OpenAM. This chapter also discusses how to prepare the DMZ for distributed authentication, deploying the Distributed Authentication service, and configuring the Distributed Authentication service.

Chapter 5, Application Authentication with Fedlets, serves as a quick primer on what Fedlets are and how to secure sites with Fedlets against OpenAM configuring Fedlets in OpenAM, and testing Fedlets in OpenAM against a Java Web Application.

Chapter 6, Implementing SAML2 Federation Patterns, serves as a quick primer on what SAML2 is and how to achieve it with OpenAM. This chapter also covers how to configure SAML Identity Providers in OpenAM and testing OpenID in OpenAM against a PHP SAML application.

Chapter 7, OAuth Authentication, serves as a quick primer on what OAuth is and how to achieve it with OpenAM. The chapter covers configuring the OAuth authorization service in OpenAM, registering OAuth clients in OpenAM, and testing OAuth in OpenAM against Facebook.

Chapter 8, Two Factor Authentication, serves as a quick primer on what two factor authentication is, as well as discussing configuring the two factor authentication module, installing a one time password token generator on Android, and integrating OpenAM with the one time password token generator.

Chapter 9, Adaptive Risk Authentication, serves as a quick primer on Adaptive Risk authentication in OpenAM. This chapter includes what Adaptive Risk authentication is, how to install Adaptive Risk authentication, what the Adaptive Risk authentication filters are, and patterns for using Adaptive Risk authentication.

What you need for this book

This book has been written on Windows 7, however, most of the instructions are equally applicable on your operating system of choice. As OpenAM is a java web application, you will need a Java Runtime installed.

Who this book is for

This book is for technical consultants who would like to become familiar with OpenAM to use for protecting their web applications. Familiarity with web application servers like Tomcat and Apache is a bonus, but not a prerequisite.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows:

"Inside the `tomcat\bin` folder, create a text file called `setenv.bat`."

A block of code is set as follows:


```
set CATALINA_OPTS=-Xmx2048m -XX:MaxPermSize=512m
```


Any command-line input or output is written as follows:

```
INFO: Deploying web application archive path-to-apache-tomcat\webapps\
openam.war
```

```
INFO: Server startup in 80846 ms
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "clicking the **Next** button moves you to the next screen".

[ Warnings or important notes appear in a box like this.]

[ Tips and tricks appear like this.]

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to feedback@packtpub.com, and mention the book title through the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/support>, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website, or added to any list of existing errata, under the Errata section of that title.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at questions@packtpub.com if you are having a problem with any aspect of the book, and we will do our best to address it.

1

Identity Management Patterns and Principles

Your interest in Identity Management is well placed – I believe Identity Management will become the next frontier of the Internet as well as the digital society. This chapter will cover the following areas:

- Defining Identity Management
- Why Identity Management is important
- How Identity Management works
- Key components of Identity Management

Defining Identity Management

I worked on the second largest Identity Management program in New Zealand, and found it a challenge to explain to people what Identity Management was. My description of "Imagine logging onto Hotmail, but without the email bit" left something to be desired. So together, let's explore the meaning of Identity Management. Wikipedia (http://en.wikipedia.org/wiki/Identity_management) describes Identity Management as:

The management of individual identifiers, their authentication, authorization, and privileges within or across system and enterprise boundaries with the goal of increasing security and productivity while decreasing cost, downtime and repetitive tasks.

I break that down into:

Understanding who someone claims to be, who they are, what they can do, and where they can do it.

How claims relate to identity

Understanding who someone claims to be is important. We all make claims or assertions about our lives. I claim to be a better blogger than I am. You might claim to be a taller person than me. Your claim is more than likely to be correct. But we need to determine whether these claims are relevant to ourselves, and then consider whether we know these claims to be correct. A more practical example of a claim is your username and password at your favorite shopping website. With these credentials, you claim to be the person who is associated with that account, and that you'd like to assert that identity. Why? So that you can continue the relationship you have with that shopping website. That's what identities are all about; identifying yourself, so you can continue that relationship where you left off. Without it, the Internet would never remember you.

Of course, you have your claims. But the next step is to verify some of those claims. We can see that your username and password are valid, but perhaps there are other parts of our relationship we want to confirm. Banks often ask secret questions, or send secret codes to verify that you know some shared secret.

Understanding identity contexts

The next step is to consider what a particular identity can do. You are you, but depending on the context, you may be a business person, someone who enjoys tennis, someone who only drinks green tea, or someone who lives in Tokyo. These contexts in turn govern what your particular identity can do. So what you can do is governed both by who you are, and the context of that relationship. This isn't a surprise, though. I have a personal YouTube account, and another for making really bad travel videos. I don't want those contexts to be associated with each other, even though I'm the same identity.

Finally, this is where you can use your identity. Some identities you have are specific to a certain context—for instance, your login ID for your computer at work isn't likely to work anywhere other than work. But your e-mail login may allow you to access other websites, such as a blogging website. In this instance, your identity is shared between different websites that have a relationship between each other like Star Trek, and are in federation with each other.

Why Identity Management is important?

But why is Identity Management important? Well, it depends on the context.

On the project I was working on, a government agency wanted someone to have a single username and password across multiple websites from different agencies. The concept of sharing the same username and password across different websites or web applications is known as **Single Sign On (SSO)**.

Examples of identity levels

So how could this single username and password work in practice? Well, let's explore the different ways in which these identities can be used:

- A random user wants to bookmark a particular part of a website, so we have to remember that user
- A known user wants to access the secure part of a website, so we have to remember that user, who they are, and what permissions they have
- The above user who wants to access the same website above, but acts in a different context, such as working for a different employer
- A user who has an identity with one website and wants to use that identity for another website

Pseudonymous identities

In the first scenario, Identity Management is important because it allows us to remember the relationship status between a system and a user. One example could be your login ID with a news website. Their primary concern is not who you are, but if you are the same person every time. This is useful because you may wish to customize the news site and remove the sports section. And in return, the site won't show you sports ads, which it can do, since it remembers your preference. This example is known as pseudonymous authentication.

Trusted identities

The next level of Identity Management is caring about the person's identity. This would be the relationship you have with your telecommunications company. They care who you are because they want to charge you for using their services. Identity Management is important in this scenario because of the financial consequences of the relationship. If someone hacked into your mobile phone account and added a whole bunch of premium services to your account, you would likely be financially responsible, or at least angry. These sorts of relationships can be thought of as community identities – identities you use in the community. For example, your friends may call you Bob, your power bill is to a Bob, but your real name is Robert. Here you have two identities, a Bob and a Robert, but only one is used in the community. That's why when signing up for some government services, they want to see evidence of the identity used in the community-on a power bill for instance.

Trusted identities with multiple contexts

A further type of Identity Management is dealing with identities in different contexts. This is important when one identity can have different permissions depending on the context they're using at the time. For example, you may be a student in one class at a university, and a tutor in another class. You're the same identity, but have a different context. This context determines what permissions you have in the course management system. As a tutor, you can mark your students. As a student, you can take tests that get marked by another tutor. But you could never take a course, and then mark it yourself, because there is no context that suits this situation.

Federated identities

Finally, you could want to take your identity with one system, and use that same identity on other systems. An example of this is using Facebook. Once you have a Facebook account, you can use your Facebook login to then associate with your Yahoo Mail account. In other words, Facebook and Yahoo have an identity federation together. Just like Star Trek, they're a collection of like-minded entities that choose to trust each other's identity systems. Identity Management is important in this context because it creates a trust relationship that allows different organizations to work together and trust common identities.

How Identity Management works

So as you can see, Identity Management is everywhere. But how does Identity Management work? Well, let's walk through the process together to think about a login ID to a system. Let's pick a banking website.

The first step is to access some secure content. Not all parts of a system are secure, for instance, the homepage of the bank website. But the Internet Banking section is secure and will ask for your identity credentials.

Entering identity credentials, such as a username or some other unique identifier and a password, is the second step. This would identify that someone knows your credentials.

The next step is to take those credentials and validate them against a directory. A directory contains a list of users and other related identity information. This list could be a SQL database, an LDAP directory, or even a flat file. Whatever it is, the credentials entered will be validated against the directory and any other authentication systems (such as the two factor authentication server, or a certificate authority). This step is known as authentication. If the credentials are correct, then it's on to the next step. If they're not correct, the system could choose to let the user re-enter the credentials, or take another security action such as locking the account.

The fourth step is authorization, which is about determining the correct permissions for the user. This depends on the context of the user, as discussed earlier in the chapter. One identity may have access to business accounts and personal accounts. These different accounts will have access to different parts of the banking website, depending on the context the identity wants to use.

For high risk transactions, a higher level of authentication assurance may be required, such as a special **One Time Password (OTP)** code sent to your mobile phone. This is known as **Two Factor Authentication (TFA)**.

The final step is accessing the secure resource, where the Identity Management system allows the user to, well, access the secure resource. This could include passing a token to any systems that the user is accessing, which describe the type of access the user has, and any other conditions, such as how long that session is valid for.

So that describes the happy path. But there's a bit more to Identity Management than that. There is registering a new account, which could be done by the user or by an administrator. There is also dealing with the exception flow, such as an incorrect password, requesting access to the system the user does not have permission to view, resetting a locked account, and resetting a user password amongst other things. Suffice it to say, these will all be touched upon in more detail in later chapters.

Key components of Identity Management

We briefly touched upon one component of an Identity Management system, the directory. But there are a few more components. Let's go through some common components and understand their purpose.

Identity Service Providers

The first component is the system with the secure resource that an identity is trying to access. This is known as a **Service Provider**. Think of this as a system that is providing a service to an identity, such as internet banking or online billing. In smaller systems, you may find the system with the secure resource to also have an Identity Management function. In fact, most systems these days have their own in-built Identity Management functions, which is fine and well, but this is the reason why some people have ten different logins to access ten different systems. And so, while it seems like a simple idea for each system to have their own Identity Management function, from a strategic perspective the total security of all the systems decreases, because people can't remember multiple usernames and passwords for multiple systems. So in larger systems, either the local Identity Management function is turned off, or was never in place.

Identity policy agents

The second component is the policy agent. A policy agent can be thought of as a gatekeeper protecting other systems that may not be compatible with an Identity Management system. A policy agent is typically tied to an infrastructure platform, such as a webserver that intercepts calls to applications, and instead redirects them to the Identity Management system for authentication and authorization. Policy agents aren't as popular as they used to be, since more systems become Identity Management-aware, and are able to communicate directly with an Identity Management system using a language such as **Security Assertion Markup Language (SAML)**.

Identity providers

The third component of the system is the authentication engine itself. For the rest of this book, we'll be referring to this component as **OpenSSO**. This system looks after the mechanics of authentication and authorization including talking to other related identity systems such as the directory, and an Identity Manager. This is known as the **Identity Provider**.

Identity data stores

The fourth component in the system is the identity data store, which can be a directory or a database. The data store holds all the identity information and is generally designed to be able to quickly find and retrieve information, rather than writing information. **Lightweight Directory Access Protocol (LDAP)** is a common method for accessing directories, which are known as LDAP directories. Active directory is an LDAP directory for those in the Windows world.

Identity managers

The fifth component in the system is an identity manager. Weirdly, this is a separate component from the authentication and authorization engine, and looks after how identities are created, related, and retired. For instance, when a user changes a password, an identity manager can distribute that password change to multiple systems so that each system stores that password. Think of an identity manager as managing identities on behalf of a lot of different systems.

Summary

In this chapter we covered what Identity Management is, why Identity Management is important, listed some examples of identity levels, how a typical Identity Management system works, and described components of an Identity Management system.

2

Installing OpenAM 10.x

In the previous chapter we talked about the broad concepts of Identity Management, including authentication. In this chapter we'll be covering how to install OpenAM 10.x, an application used to do authentication. We'll be doing this installation on a Windows 7 Home Premium 64-bit edition machine. The topics this chapter will cover are:

- Downloading OpenAM 10.x
- Prerequisites for OpenAM 10.x
- Installing OpenAM 10.x

Downloading OpenAM 10.x

The first step is to download OpenAM. Like most open source products, there are two different flavors:

- OpenAM
- OpenAM Enterprise

So what's the difference? The OpenAM Enterprise Download Stack page <http://forgerock.com/download-stack/> states:

ForgeRock's Enterprise products are fully-productized, off-the-shelf IAM solutions that are developed, tested and supported by the ForgeRock engineering, services and support teams. Enterprise products are released under a commercial license that allow developers to freely use them in development environments, POCs or to simply "kick-the-tires." For production use a subscription is required.

In effect, ForgeRock, the company that maintains OpenAM, have turned the open source code for OpenAM into a precompiled binary that is freely available for anyone to download and use in a non-production manner. However, anyone is available to take the OpenAM source code and compile their own binaries to use as they wish. The choice is theirs.



To Open Source or not to Open Source?

While it may appear cheaper to build your own binaries from source code, there are other expenses when it comes to software systems, such as support, maintenance, and compatibility. Do consider these issues, and evaluate mitigating risk by purchasing an Enterprise Subscription if suitable.

Now that we're aware of the different types of OpenAM, we're going to be using the OpenAM Enterprise 10.1.0 stack available from <http://forgerock.com/openam-downloads/>. Click on the ZIP link to download the file from http://download.forgerock.org/downloads/enterprise/openam/openam10/10.1.0/openam_10.1.0.zip.



What's the difference between the WAR and the ZIP file?

The WAR file contains only the deployable OpenAM file. The ZIP file contains all information about the release, including the source code. If in doubt, download the ZIP file.

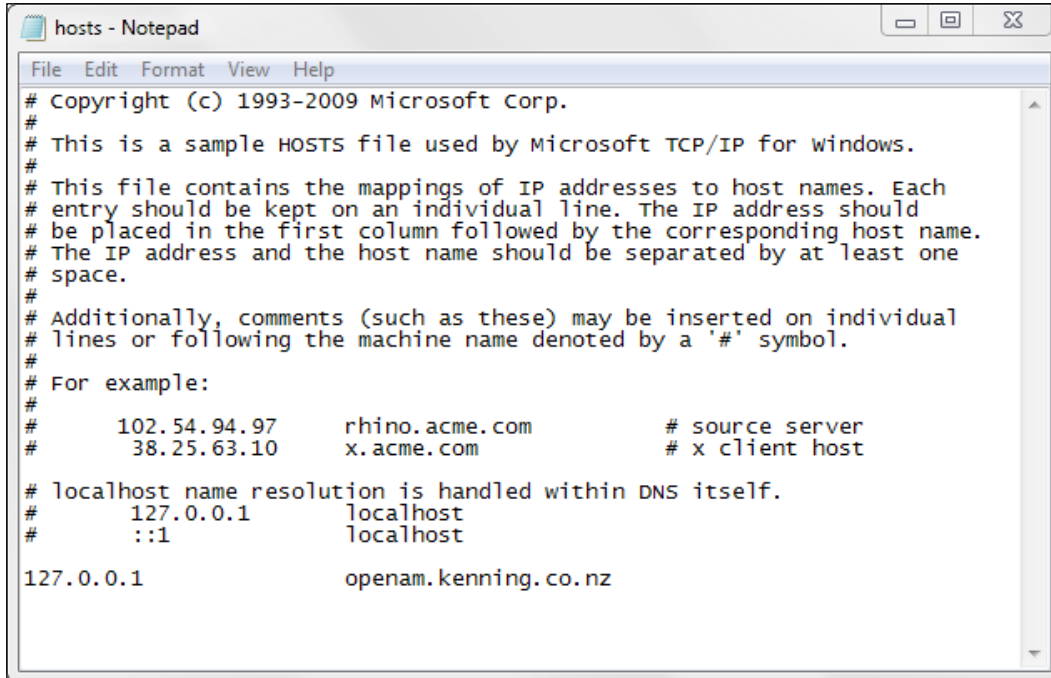
Prerequisites for OpenAM

While we wait for OpenAM to download, let's discuss the OpenAM prerequisites. One advantage to using the Enterprise version of OpenAM is access to the OpenAM Install Guide available from <http://docs.forgerock.org/en/openam/10.1.0/OpenAM-10.1.0-Install-Guide.pdf>.

Creating a fully qualified domain name

1. Click on the **Start** button, find **Notepad**, right-click on it, and select **Run as Administrator**. This is important because if you try and edit system files as a normal user, you will get a permission denied error.
2. Open the **hosts** file in the notepad by navigating to `C:\Windows\System32\drivers\etc\hosts` file. If you don't see it, you'll need to change the dropdown in the notepad from **Text Documents (*.txt)** to **All Files (*.*)**.

3. Once it is opened it will look like the following screenshot:



```

File Edit Format View Help
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com          # source server
#       38.25.63.10       x.acme.com              # x client host
#
# localhost name resolution is handled within DNS itself.
#       127.0.0.1        localhost
#       ::1              localhost
127.0.0.1                openam.kenning.co.nz

```

4. On a new line, type your computer's IP address (which can be the localhost address – 127.0.0.1) and a domain name (I'm using openam.kenning.co.nz).
5. From the **File** menu, click on **Save** and then close the file.
6. You should now be able to ping the address openam.kenning.co.nz.



Make sure your domain has at least a domain and a subdomain; otherwise you'll run into trouble when configuring cookie domains during the OpenAM install.

Installing the Java Runtime Environment

OpenAM requires Java, either the **Java Runtime Environment (JRE)** or the **Java Software Development Kit (JDK)**. These days, most computers will have the JRE installed, otherwise it can be downloaded from <http://www.java.com/en/download/index.jsp>.

For OpenAM 10.1, the version must be greater than 1.6.0_10. To find this out, type the following on a command prompt:

```
java -version
```

This returns Java version 1.6.0_25 on the machine.

Downloading the Tomcat application server

As a Java web application, OpenAM can be deployed in a variety of Java web application servers. For OpenAM 10.1.0, the supported web application containers can be found in the OpenAM 10.1.0 release notes available at <http://docs.forgerock.org/en/openam/10.1.0/release-notes/>.

For our prototype, we'll be using the latest version of Apache Tomcat 6.0, which can be downloaded from <http://tomcat.apache.org/download-60.cgi>.

Extract Tomcat to a directory, such as C:\tomcat.

Configuring Tomcat for OpenAM

Tomcat requires changes to its configuration files to support OpenAM, specifically increasing the minimum JVM heap size to at least 1024 MB, and the permanent generation size to at least 256 MB.

1. Inside the tomcat\bin folder, create a text file called setenv.bat. Add the following line:

```
set CATALINA_OPTS=-Xmx2048m -XX:MaxPermSize=512m
```

This configures the maximum amount of memory that various parts of Java can use. The minimum recommended is 1024m and 256m respectively.

2. Copy openam-server-10.1.0-Xpress.war to the webapps folder present at C:\tomcat\webapps, and rename it as openam.war.
3. Next, run startup.bat. A Java command window should appear showing the startup of Tomcat. Look for the lines:

```
INFO: Deploying web application archive path-to-apache-tomcat\
webapps\openam.war
```

```
INFO: Server startup in 80846 ms
```

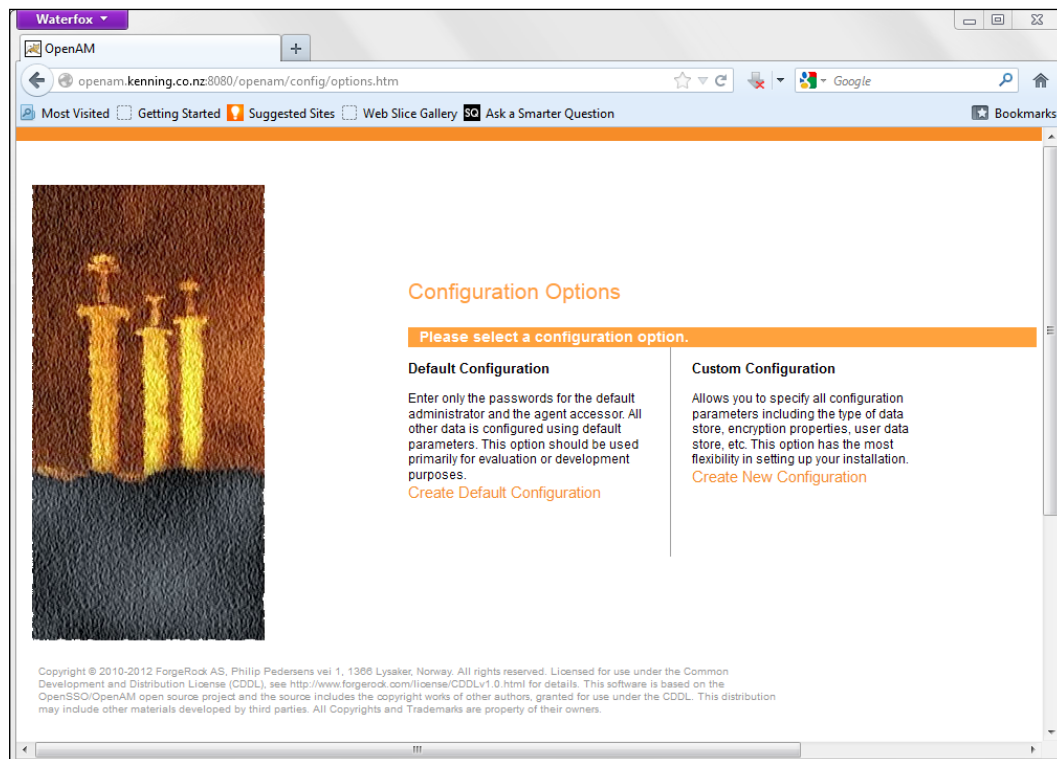
4. In a web browser, go to <http://openam.kenning.co.nz:8080/openam> (of course, substituting your domain name set up earlier in the chapter).



It doesn't work! What do I do?

While researching for this book, I encountered all types of errors. The first step is to explore what the Java command window says. Often after configuring OpenAM, I would encounter **out of memory** errors, which the Java command window stated. A second step is to try an alternate configuration. Originally I tried Tomcat 7, but the OpenAM console would not load. After reverting to Tomcat 6, the OpenAM console loaded perfectly. Finally, have a look inside the Tomcat logs folder. Often, useful information will appear in the logs regarding configuration errors, especially about syntax.

Installing OpenAM 10.1.0



There are two configuration options, default and custom, as shown in the preceding screenshot. Default is used for setting up a prototype environment fast, and only requires an admin account password and an agent account password.

Custom is a bit more detailed, so let's explore those options:

1. On the first screen, enter the password for the default user, amAdmin, and then click on **Next**. You will then see a screen as shown in the following screenshot:

The screenshot shows the 'OpenAM Configurator' window with the 'Custom Configuration Option' tab selected. The left sidebar lists the configuration steps: 1. General, 2. Server Settings (selected), 3. Configuration Store, 4. User Store, 5. Site Configuration, 6. Agent Information, and 7. Summary. The main area is titled 'Step 2: Server Settings' and includes the instruction 'Confirm the following settings to use for the server.' and a note '* Indicates required field'. The 'Server Settings' section contains four fields: '* Server URL' with the value 'http://openam.kenning.co.nz:8080', '* Cookie Domain' with the value '.kenning.co.nz', '* Platform Locale' with the value 'en_US', and '* Configuration Directory' with the value 'C:/Users/Waylon/openam'. At the bottom of the window are 'Previous', 'Next', and 'Cancel' buttons.

- **Server URL** refers to the URL of the OpenAM server, and by default populates itself with the URL you are using to access the site. This could be different if you had set up a reverse proxy in the front, or if you're using a different port.
- **Cookie Domain** refers to the domain that is used for the OpenAM session cookie, and will populate with the domain or subdomain in the Server URL, excluding the server name. Since my server name was openam.kenning.co.nz, the cookie domain will be .kenning.co.nz.

- **Platform Locale** by default is set to `en_US`, and we'll leave that for now.
 - **Configuration Directory** is where the confirmation information for OpenAM should reside.
2. Click on **Next**. You will then see a screen as shown in the following screenshot:

The screenshot shows the 'OpenAM Configurator' window with the 'Custom Configuration Option' tab selected. The left sidebar lists steps: 1. General, 2. Server Settings, 3. Configuration Store (selected), 4. User Store, 5. Site Configuration, 6. Agent Information, and 7. Summary. The main area is titled 'Step 3: Configuration Data Store Settings' and includes instructions: 'If no other OpenAM instance already exists in the environment, then choose First Instance. If one or more OpenAM instances already exist in the environment, choose Add to Existing Deployment.' Below this are two radio buttons: 'First Instance' (selected) and 'Add to Existing Deployment?'. A note indicates that an asterisk (*) denotes a required field. The 'Configuration Store Details' section contains the following fields:

Configuration Store Details	
Configuration Data Store	<input checked="" type="radio"/> OpenAM <input type="radio"/> OpenDJ or Oracle Directory Server Enterprise Edition
* SSL/TLS Enabled	<input type="checkbox"/>
* Host Name	<input type="text" value="localhost"/>
* Port	<input type="text" value="50389"/>
* Admin Port	<input type="text" value="4444"/>
* JMX Port	<input type="text" value="1689"/>
* Encryption Key	<input type="text" value="e/PDdCLV5b/SgmEyPYHZ/yq8y1Ba"/>
* Root Suffix	<input type="text" value="dc=openam,dc=forgerock,dc=org"/>


At the bottom of the window are three buttons: 'Previous', 'Next' (highlighted), and 'Cancel'.

- **Configuration Data Store** refers to where you want to store the configuration data in a directory. Your choice is either to store it in the embedded directory server of **OpenAM**, or an **OpenDJ** or **Oracle Directory Server Enterprise Edition** server if you have one available. Since we're setting up a prototype, we'll just pick the **OpenAM** option.

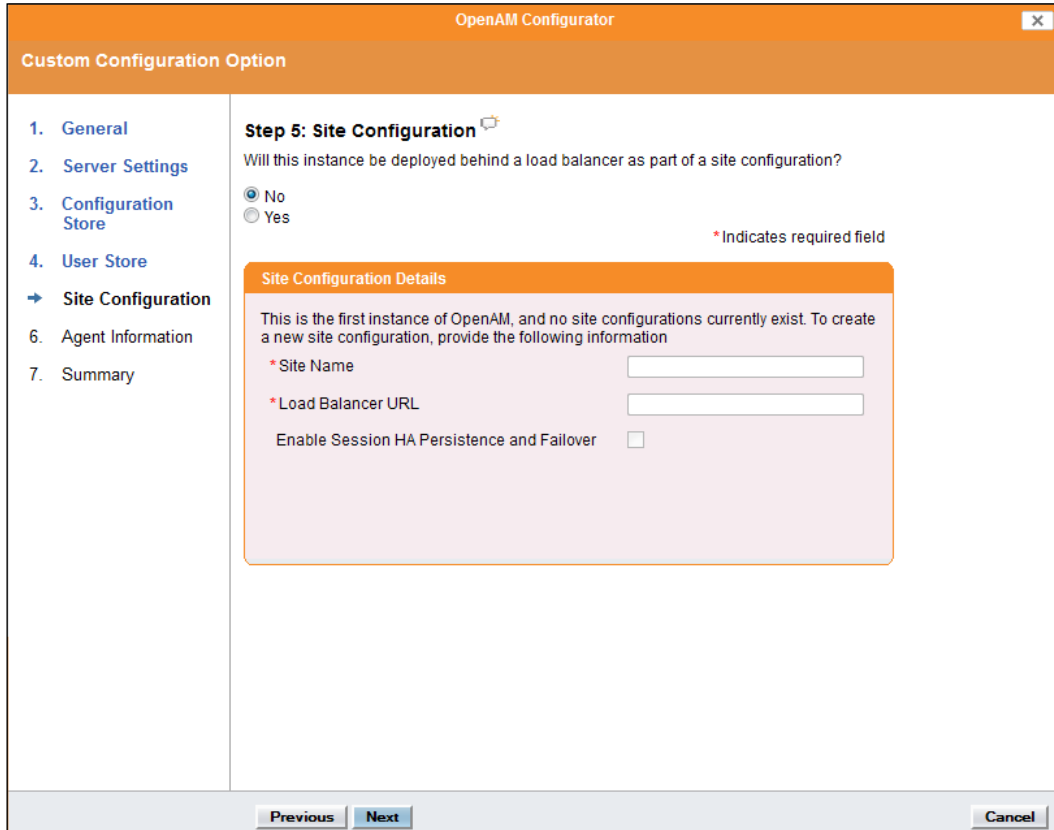
- If you select **OpenAM**, some settings will be disabled, such as whether the directory is SSL/TLS enabled, and what the hostname of the directory server is, and so on. There are settings for **Port**, **Admin Port**, **JMX Port**, **Encryption Key**, and **Root Suffix** which all revolve around configuration settings for the directory server.
3. For our prototype we can leave them as they are and click on **Next**. You will then see a screen as shown in the following screenshot:

The screenshot shows the 'OpenAM Configurator' window with the title 'Custom Configuration Option'. On the left is a navigation pane with steps 1 through 7. Step 4, 'User Store', is selected and highlighted with a blue arrow. The main area is titled 'Step 4: User Data Store Settings' and includes a help icon. Below the title is a paragraph explaining the data store options. There are two radio buttons: 'OpenAM User Data Store' (unselected) and 'Other User Data Store' (selected). A red asterisk indicates required fields. Below this is a 'User Store Details' section with several fields: 'User Data Store Type' (radio buttons for Oracle Directory Server Enterprise Edition, OpenDJ, Active Directory with Host and Port, AD with Domain Name, Active Directory Application Mode, and IBM Tivoli Directory Server), 'SSL/TLS Enabled' (checkbox), 'Directory Name' (text field with 'openam.kenning.co.nz'), 'Port' (text field with '389'), 'Root Suffix' (text field with 'dc=openam,dc=forgerock,dc=org'), 'Login ID' (text field with 'cn=Directory Manager'), and 'Password' (text field). At the bottom are 'Previous', 'Next', and 'Cancel' buttons.

4. The **User Data Store Type** refers to where user information will be stored. In a production environment, this should be configured to use a robust LDAP directory, such as **OpenDJ**, **Oracle Directory Server Enterprise Edition**, or **Active Directory with Host and Port**, and so on. If so, various options on how to connect to that directory are required. However, since we're creating a prototype, we can select the **OpenAM User Data Store**.

 Don't use the OpenAM User Data Store in a Production environment.

5. Click on **Next**. You will then see a screen as shown in the following screenshot:



The screenshot shows the 'OpenAM Configurator' window with the title 'Custom Configuration Option'. On the left is a sidebar with a list of steps: 1. General, 2. Server Settings, 3. Configuration Store, 4. User Store, 5. Site Configuration (highlighted with a blue arrow), 6. Agent Information, and 7. Summary. The main area is titled 'Step 5: Site Configuration' and contains the question 'Will this instance be deployed behind a load balancer as part of a site configuration?'. Below this are two radio buttons: 'No' (selected) and 'Yes'. A red asterisk with the text '* Indicates required field' is shown. Below the radio buttons is a box titled 'Site Configuration Details' which contains the text: 'This is the first instance of OpenAM, and no site configurations currently exist. To create a new site configuration, provide the following information'. Inside this box are two required fields: '* Site Name' and '* Load Balancer URL', each with a text input box. Below these fields is a checkbox labeled 'Enable Session HA Persistence and Failover'. At the bottom of the window are three buttons: 'Previous', 'Next' (highlighted), and 'Cancel'.

6. If OpenAM is behind a load balancer it needs to be made aware of this, specifically if it needs to be configured for high availability.
7. For our prototype we can click on **Next**.



Do I need High Availability?

Like all good features it comes with a cost -- complexity. High availability requires essentially a mirror environment so that if one component fails, the end user will not notice the impact. However, OpenAM is just one component in a chain between the browser and the web servers. Each of those components needs to be engineered for high availability to ensure users aren't impacted by failure.

- Next, enter the **Policy Agent** password and then click on **Next**. You will then see a screen as shown in the following screenshot:

The screenshot shows the 'OpenAM Configurator' window with the 'Custom Configuration Option' tab selected. The left sidebar lists the configuration steps: 1. General, 2. Server Settings, 3. Configuration Store, 4. User Store, 5. Site Configuration, 6. Agent Information, and a selected 'Summary' option. The main content area is titled 'Configurator Summary Details' and includes a warning: 'Take a moment to review the settings below. If any values are incorrect you may go back and modify the settings prior to configuration.' Below this, there are three sections: 'Configuration Store Details' with fields for SSL/TLS Enabled (No), Host Name (localhost), Listening Port (50389), Root Suffix (dc=openam,dc=forgerock,dc=org), User Name (cn=Directory Manager), and Directory Name (C:/Users/Waylon/openam); 'User Store Details' (Using Configuration Store Settings); and 'Site Configuration Details' (This instance is not setup behind a load balancer). At the bottom, there are 'Previous', 'Create Configuration', and 'Cancel' buttons.

- Finally, a **Configurator Summary Details** page will appear, as shown in the preceding screenshot, that will allow you to review your details. If they're correct, click on the **Create Configuration** button to start the configuration.
- When the configuration is complete, you should see a **Configuration Complete** dialog box and a link to login to OpenAM.

Summary

In this chapter we downloaded OpenAM 10.x, we installed and configured the prerequisites required for OpenAM 10.x, and thus installed OpenAM 10.x.

3

Cross-Domain Single Sign On

In the last chapter we covered installing OpenAM, which is great, but the next step is to start securing multiple sites in multiple domains. This chapter will cover the following points:

- Securing a web server on the same domain as OpenAM
- Securing an application server on a different domain to OpenAM

An introduction to Cross-Domain Single Sign On

Cross-Domain Single Sign On (CDSSO) is a feature of OpenAM that allows authentication to go between different domains. When OpenAM was installed in our prototype, it was installed against `openam.kenning.co.nz`, which means that OpenAM will function correctly against any `.kenning.co.nz` domain because the OpenAM configurator sets the cookie domain to `.kenning.co.nz`. But what if you own other domain names, such as `.kenning.local`? We'll need to enable Cross-Domain Single Sign On to allow the other domain names to be protected by OpenAM.

So how is Cross-Domain Single Sign On different from Federation? Cross-Domain Single Sign On is about one instance of OpenAM securing sites on different domains. Federation is about OpenAM trusting other Identity Management systems operating on different domains. If the other domain is already secured, use Federation. If the other domain has no security, use Cross-Domain Single Sign On.

Securing an Apache 2.4 local domain website

The first step we want to take is to secure an installation of Apache 2.4 operating on the same domain as OpenAM. OpenAM is running at `http://openam.kenning.co.nz:8080/openam`, and our installation of Apache 2.4 will run at `http://apache.kenning.co.nz`. To do this, we'll need to create an Agent profile in OpenAM, and then download and install the Apache Policy Agent.

We'll be using some new domain names in this chapter, so as per the instructions in the previous chapter, open your host file, create the domain names `apache.kenning.co.nz`, and `tomcat.kenning.local`, and point them both to `127.0.0.1`.

Creating an Apache Policy Agent profile in OpenAM

In this section, we will create an Apache Policy Agent profile in OpenAM that will define the settings for the site we will be protecting.

1. Log into OpenAM and click on the **Access Control** button at the top of the page.
2. Next, click on the realm we want to protect. In our prototype, there's only one realm, called **Top Level Realm**.
3. Click on the **Agents** button at the far right of the page.
4. Click on the **New Agent** button under the **Web** tab.
5. Give the agent a name. We'll call ours `Apache24`.
6. Give the agent a password. We'll use `apachepassword`.
7. Enter the OpenAM server URL. We'll use `http://openam.kenning.co.nz:8080/openam`.
8. Enter the Apache URL we'll be protecting, including the port number (the default being 80). We'll use `http://apache.kenning.co.nz:80`.
9. You can choose whether you want the configuration to be stored within the agent or within the OpenAM server. I prefer storing configurations within the OpenAM server, so we'll select the centralized option.

Waterfox

OpenAM

openam.kenning.co.nz:8080/openam/agentconfig/Agents

VERSION

User: amAdmin Server: Waylon-PC

LOG OUT

New Web

Create Cancel

* Indicates required field

* Name: Apache24

* Password:

* Re-Enter Password:

Configuration: ☐ Local ☒ Centralized
Where agent properties are stored. Local is the server on which the agent is running. Centralized is the OpenAM Server

* Server URL: http://openam.kenning.co.nz:8080/openam
protocol://host:port/deploymentUri e.g. http://opensso.sample.com:58080/opensso

* Agent URL: http://apache.kenning.co.nz:80
protocol://host:port e.g. http://agent1.sample.com:1234

10. Click on the **Create** button.
11. Now we need to go into the agent profile we just created and configure it to only do authentication (or Single Sign On) and not authorization. This is because we haven't defined any authorization roles yet, which means we wouldn't be able to access resources (such as pages).
12. Click on the agent profile we just created, **Apache24**.
13. Scroll down and tick the **SSO Only** mode, which will force Single Sign On only, and not apply authorization.
14. Click on the **Save** button.

Securing Apache with the OpenAM Policy Agent

We now need to secure our Apache installation with the OpenAM Policy Agent. This step presumes you have an Apache web server installed. If not, XAMPP is a great way to get Apache installed on Windows, and can be downloaded from <http://www.apachefriends.org/en/xampp-windows.html>. My Apache install was located at `C:\xampp\`.

1. The first step is to download the Apache Policy Agent. I'm using Apache 2.4, so I'll download the Apache 2.4 Policy Agent. All the Community OpenAM Policy Agents can be found at <http://forgerock.org/openam.html>. The Enterprise Release OpenAM Policy agents can be found at <https://download.forgerock.com/#/openam>. The specific policy agent download I used was http://download.forgerock.org/downloads/openam/webagents/nightly/WINNT/apache_v24_WINNT_agent_3.1.0-Xpress.zip. This is a nightly build, so you should refer to the latest policy agent download link available from the websites above.
2. Next, extract the download to a folder. We should extract it to `C:\xampp\apache24_agent\`.
3. We now need to create a text file which will contain the password for the agent, which we created when we created the Agent profile. Create a file called `C:\xampp\apache24_agent\apacheagentpassword.txt` and save the password `apachepassword`.
4. Open a command line. Navigate to the `C:\xampp\apache24_agent\bin\` folder. Now, type `agentadmin.bat --install` which will start the Agent installation process.
5. In the agent installation window, enter the path of your Apache configuration folder; in my case it was `C:\xampp\apache\conf`.
6. Next, enter the URL to your OpenAM server. Mine was `http://openam.kenning.co.nz:8080/openam`.
7. Now enter the URL to your Apache web server that the agent is protecting, including the port number. Mine was `http://apache.kenning.co.nz:80`.
8. Enter the agent profile name that we created earlier, `Apache24`.
9. Type the path to the text file that contains the agent password. Mine was `C:\xampp\apache24_agent\apacheagentpassword.txt`.
10. You're now given a chance to review the options. Once done, press `1` to continue with the installation.

11. Because our Apache web server wasn't started earlier, do that now and head to `http://apache.kenning.co.nz`.
12. We should get redirected to the OpenAM login page. Enter your OpenAM account details (preferably using the testing account with the username `demo` and password `changeit`, or you could use the administrator details with the username `amAdmin` and password `adminpass`). We should get redirected back to our Apache content.

Securing a Tomcat 6 remote domain website

In this section, we'll create another instance of Tomcat, create a Tomcat Policy Agent profile, install the Tomcat Policy Agent, and configure it for Cross-Domain Single Sign On, as per the Java EE Policy Agent Installation Guide that can be accessed at <http://openam.forgerock.org/openam-documentation/openam-doc-source/doc/jee-install-guide/index.html>.

Configuring Tomcat and creating a Tomcat Policy Agent profile

In this step, we're going to be installing another instance of Tomcat, different from the instance running OpenAM. We'll run this instance on a different domain, `tomcat.kenning.local`.

1. Extract the Tomcat 6 download file. I extracted it to `C:\tomcat.local`.
2. We need to change the ports so they don't clash with the existing instance of Tomcat. Open `C:\tomcat.local\conf\server.xml` in a text editor.
3. Change all the ports from the 8000 range to 9000 range, so that the main port goes from 8080 to 9080 for instance. Save the file.
4. From the command line, run `C:\tomcat.local\bin\startup.bat` and see if Tomcat starts.
5. In a browser, head to `http://tomcat.kenning.local:9080/docs/` and you should see the Tomcat documentation site.
6. Shutdown Apache by running `C:\tomcat.local\bin\shutdown.bat`.
7. In a browser, log in to **OpenAM** and click on the **Access Control** button.
8. Next, click on the **Top Level Realm** link.
9. Click on the **Agents** button.

10. Click on the **J2EE** tab.
11. Click on the **New Agent** button.
12. Give the Tomcat agent a name. We'll call ours Tomcat6.
13. Give the Tomcat agent a password. We'll use tomcatpassword.
14. Enter the OpenAM server URL. We'll use `http://openam.kenning.co.nz:8080/openam`.
15. Enter the URL to the J2EE application that the agent protects, including the port number (the default being 80) and the protected resource (or sub-directory). We'll use `http://tomcat.kenning.local:9080/docs`.

The screenshot shows a web browser window titled 'Waterfox' with a single tab for 'OpenAM'. The address bar shows the URL 'openam.kenning.co.nz:8080/openam/agentconfig/Ag...'. The page header includes a 'VERSION' button, a 'LOG OUT' button, and user information: 'User: amAdmin Server: Waylon-PC'. The main content area is titled 'New J2EE' and contains the following fields and options:

- Name:** Text input field containing 'Tomcat6'.
- Password:** Password input field (masked with dots).
- Re-Enter Password:** Password input field (masked with dots).
- Configuration:** Radio buttons for 'Local' (selected) and 'Centralized'. Below the buttons is a note: 'Where agent properties are stored. Local is the server on which the agent is running. Centralized is the OpenAM Server'.
- Server URL:** Text input field containing 'http://openam.kenning.co.nz:8080/openam'. Below the field is a hint: 'protocol://host:port/deploymentUri e.g. http://opensso.sample.com:58080/opensso'.
- Agent URL:** Text input field containing 'http://tomcat.kenning.local:9080/docs'. Below the field is a hint: 'protocol://host:port/deploymentUri e.g. http://agent1.sample.com:1234/agentapp'.

At the top right of the form are 'Create' and 'Cancel' buttons. A red asterisk with the text '* Indicates required field' is located to the right of the 'Create' button.

16. Click on the **Create** button.
We now need to configure this agent to only do Single Sign On (or authentication), rather do authorization, since we still haven't created any authorization roles.
17. Go back to the newly created **Agent** settings.

18. Under the **General** section, under the **Agent Filter** mode, select ALL in the **Current Values** box, and click on the **Remove** button.
19. In the **Corresponding Map Value** box, type `SSO_ONLY`, and click on the **Add** button.



What are hot swappable values?

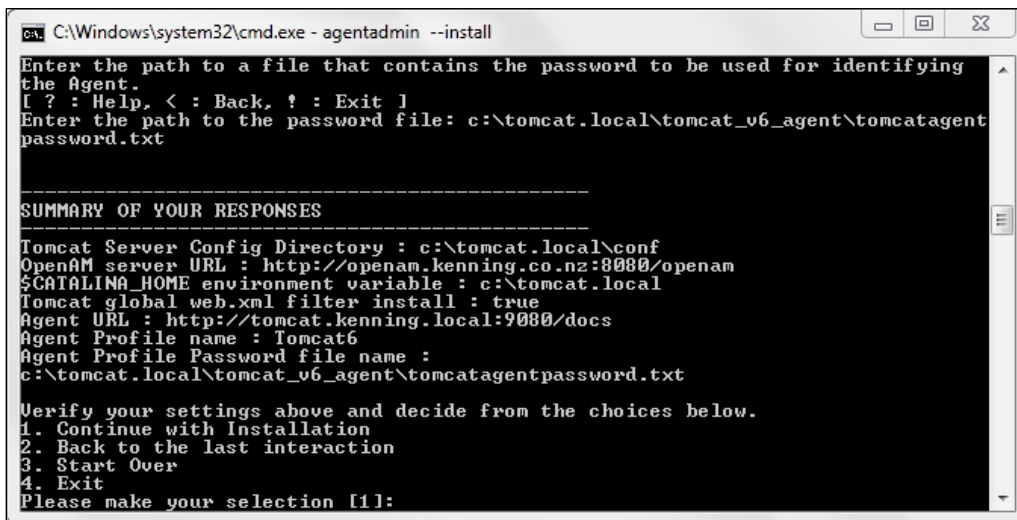
Hot swappable values are settings that can be changed without having to restart the web server or application server. The above value was not hot swappable, so for it to apply, we would have to restart the Tomcat server that has this policy agent instance installed.

20. Click on the **Save** button.

Securing Tomcat with the OpenAM Policy Agent

1. Download the Tomcat Policy Agent. All the OpenAM Policy Agents can be found at <http://forgerock.org/openam.html>. The Enterprise Release OpenAM Policy Agents can be found at <https://download.forgerock.com/#/openam>.
2. The specific policy agent download I used was http://download.forgerock.org/downloads/openam/j2eeagents/nightly/tomcat_v6_agent_3.1.0-Xpress.zip. This is a nightly build, so you should refer to the latest policy agent download link available from the websites mentioned earlier.
3. Extract the folder contents. I extracted to `C:\tomcat.local\tomcat_v6_agent`.
4. Once again, we need to create a text file which will contain the password for the agent, which we created when we created the agent profile. Create a file called `C:\tomcat.local\tomcat_v6_agent\tomcatagentpassword.txt` and save the password `tomcatpassword`.
5. Open a command line. Navigate to the `C:\tomcat.local\tomcat_v6_agent\bin\` folder. Now, type `agentadmin.bat --install` which will start the agent installation process.
6. Enter the path to the Tomcat configuration folder. In my case, this was `C:\tomcat.local\conf`.
7. Next, enter the URL to your OpenAM server. Mine was `http://openam.kenning.co.nz:8080/openam`.

8. Enter the URL to your Tomcat installation, known as \$CATALINA_HOME. Ours should be C:\tomcat.local.
9. The agent will ask to install the agent filter in the global web.xml file. Press *Enter*, as true was already selected.
10. Enter the agent URL including the deployment folder. In my case, this was http://tomcat.kenning.local:9080/docs.
11. Enter the agent profile name we created earlier, Tomcat6.
12. Enter the path to the password file we created earlier, which should ideally be C:\tomcat.local\tomcat_v6_agent\tomcatagentpassword.txt.



```
C:\Windows\system32\cmd.exe - agentadmin --install

Enter the path to a file that contains the password to be used for identifying
the Agent.
[ ? : Help, < : Back, ! : Exit ]
Enter the path to the password file: c:\tomcat.local\tomcat_v6_agent\tomcatagent
password.txt

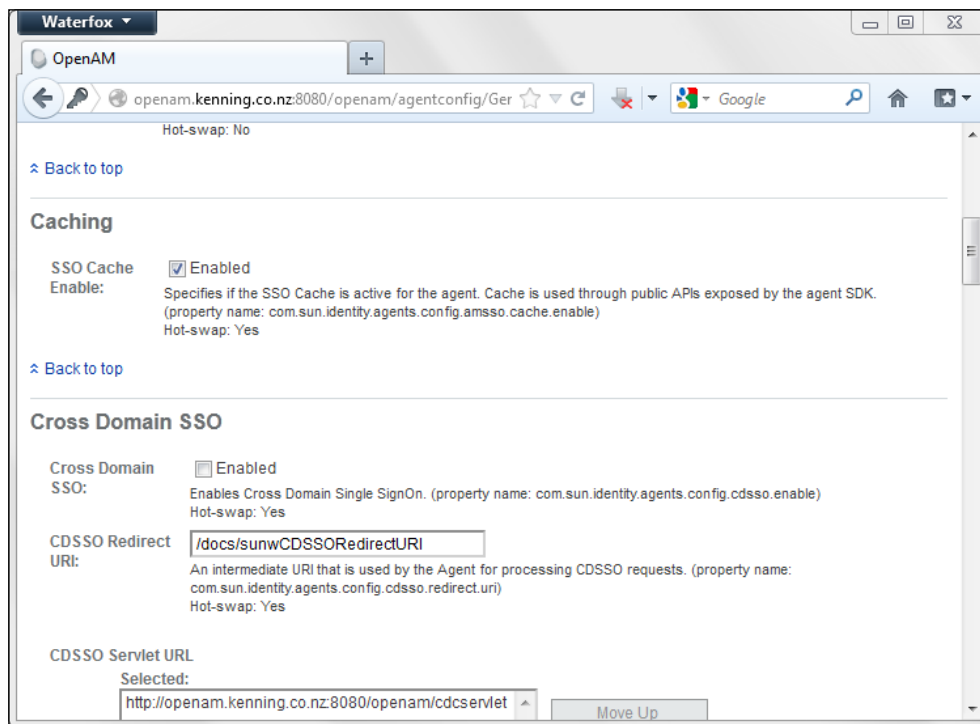
-----
SUMMARY OF YOUR RESPONSES
-----
Tomcat Server Config Directory : c:\tomcat.local\conf
OpenAM server URL : http://openam.kenning.co.nz:8080/openam
$CATALINA_HOME environment variable : c:\tomcat.local
Tomcat global web.xml filter install : true
Agent URL : http://tomcat.kenning.local:9080/docs
Agent Profile name : Tomcat6
Agent Profile Password file name :
c:\tomcat.local\tomcat_v6_agent\tomcatagentpassword.txt

Verify your settings above and decide from the choices below.
1. Continue with Installation
2. Back to the last interaction
3. Start Over
4. Exit
Please make your selection [1]:
```

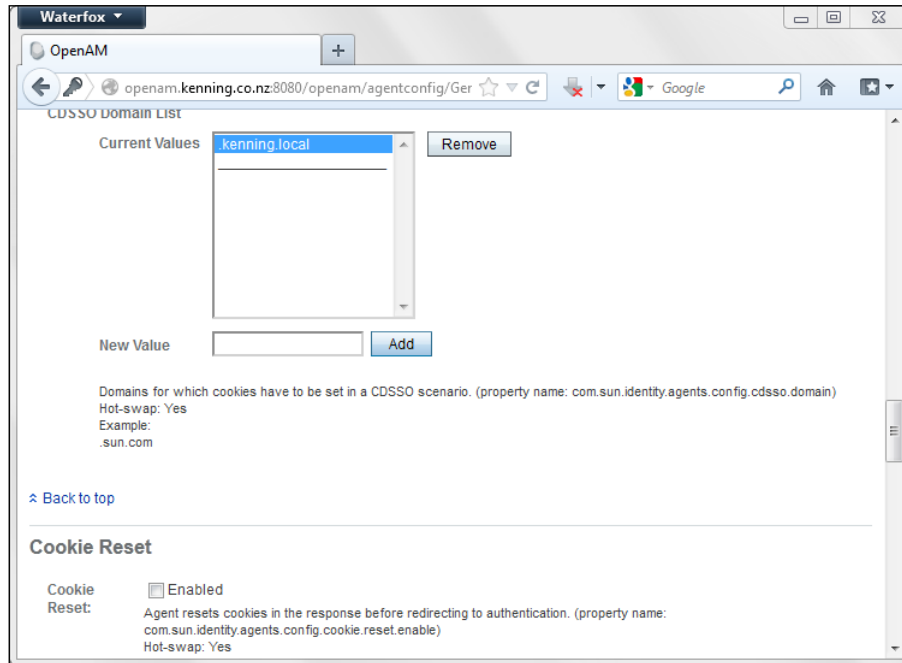
13. Review your configuration settings, and if you agree, type 1 to continue the install.
Our installation should be done, with the changes detailed in C:/tomcat.local/tomcat_v6_agent/installer-logs/audit/install.log.
14. Since this is the first time we will be using this Tomcat instance, start Tomcat from the command line by running C:\tomcat.local\bin\startup.bat.
15. In a browser, try and access http://tomcat.kenning.local:9080/docs. After logging in, your browser should detect infinite loops and not display a page. This is because Cross-Domain Single Sign On has not been enabled.

Configuring a Tomcat Agent profile for Cross-Domain Single Sign On

1. In a browser, log in to **OpenAM** and click on the **Access Control** button.
2. Next, click on the **Top Level Realm** link.
3. Click on the **Agents** button.
4. Click on the **J2EE** tab.
5. Select the **Tomcat6** agent profile that we created earlier.
6. Click on the **SSO** tab.



7. Tick the **Enabled** box for the **Cross Domain SSO** option.



8. In the **CDSSO Domain List** section, in the **New Value** box, add `.kenning.local` (note the leading dot). In this step, we're saying that all sessions from a `kenning.local` domain name are valid, because the agent should create the session cookie with this cookie domain.
9. Click on the **Save** button.
10. In a browser, try and access `http://tomcat.kenning.local:9080/docs`. After logging in, your browser should redirect you to the Tomcat documentation site.

Summary

In this chapter we created two sites: one Apache based and the other Tomcat based, and secured them with Policy agents using authentication only. The Tomcat site was on a different domain, so we enabled Cross-Domain Single Sign On to enable OpenAM to secure that site. In the next chapter, we'll be covering distributed authentication, which allows us to protect our OpenAM installation and expose services using a component deployed to the DMZ.

4

Distributed Authentication

In the last chapter we covered cross-domain authentication, where policy agents talk to OpenAM to verify authentication. But how do we protect OpenAM from exposure? In this chapter we'll:

- Install distributed authentication on a separate DMZ server to protect OpenAM from direct access
- Configure a distributed authentication server
- Configure a distributed authentication application
- Test a distributed authentication

Understanding distributed authentication

Distributed authentication revolves around two key concepts: how policy agents work and a defense-in-depth architecture.

How policy agents communicate with OpenAM

When a user accesses a URL that is protected by a **Policy Agent**, the policy agent will intercept that request to the application or web server. It will examine the request and look for information to see if that user is already authenticated and authorized. Typically that information will be inside a cookie or a session variable.

If the user is not already authenticated, the policy agent will force a redirect from the protected URL to the OpenAM login page. The user then accesses the OpenAM login page, enters their credentials, and is then redirected back to the protected URL. The policy agent will once again look at the request will see a valid credential to log in, and will allow the request to reach the application or web server.

But in all of this, there is a relationship between the policy agents and OpenAM. How does the policy agent know that a credential is valid? It asks OpenAM.

While this is useful for the policy agents, exposing OpenAM to the world increases the likelihood of it being targeted for vulnerabilities and attacks.

Understanding defense-in-depth architectures

Defense-in-depth architectures refers to having multiple different barriers between systems. For instance, a user computer could be directly connected to the Internet, as could a web server. But either of these machines could be hacked through known or unknown vulnerabilities, misconfigurations, or other typical security exploits.

That's why for systems connecting across organization boundaries (or to the Internet), there is often the concept of a **Demilitarized Zone (DMZ)**. This is a lower trust network from the primary network separated through the use of Firewalls and other networking devices. This is the same concept as protecting a castle with both a moat and wall and a door. Even if one barrier is crossed, there are still multiple other different barriers.

So when a user tries to authenticate against OpenAM, their request would first hit a Firewall or other networking device, which would inspect the request and allow it if it met certain criteria. Then the request would be served by an application server in a DMZ which would inspect the request. If the request was valid, the DMZ application server would talk through another Firewall to OpenAM on the primary network.

Preparing OpenAM for distributed authentication

The first step is to configure the cookie domains and realms within OpenAM. This is especially important if using different domain names.

1. Log in to OpenAM.
2. Next click on the **Configuration** tab, then the **System** subtab, and finally click on the **Platform** link.

The following screenshot shows the cookie domains configured in OpenAM, and is the location within OpenAM where you would configure for alternate or additional cookie domains.

Waterfox

OpenAM

openam.kenning.co.nz:8080/openam/service/SCPlatform

VERSION

LOG OUT

User: amAdmin Server: Waylon-PC

OpenAM

Platform Save Reset Back to Service Configuration

Global Attributes

Platform Locale: en_US
The default locale.

Cookie Domains

Current Values .kenning.co.nz Remove

New Value Add

The list of domains into which OpenAM cookies will be written.

- Under **Cookie Domains**, make sure that the domain you're using is listed as a domain. Since my distributed authentication server will be called `dmz.kenning.co.nz`, which is still the same as my OpenAM domain, I don't need to change the values here.

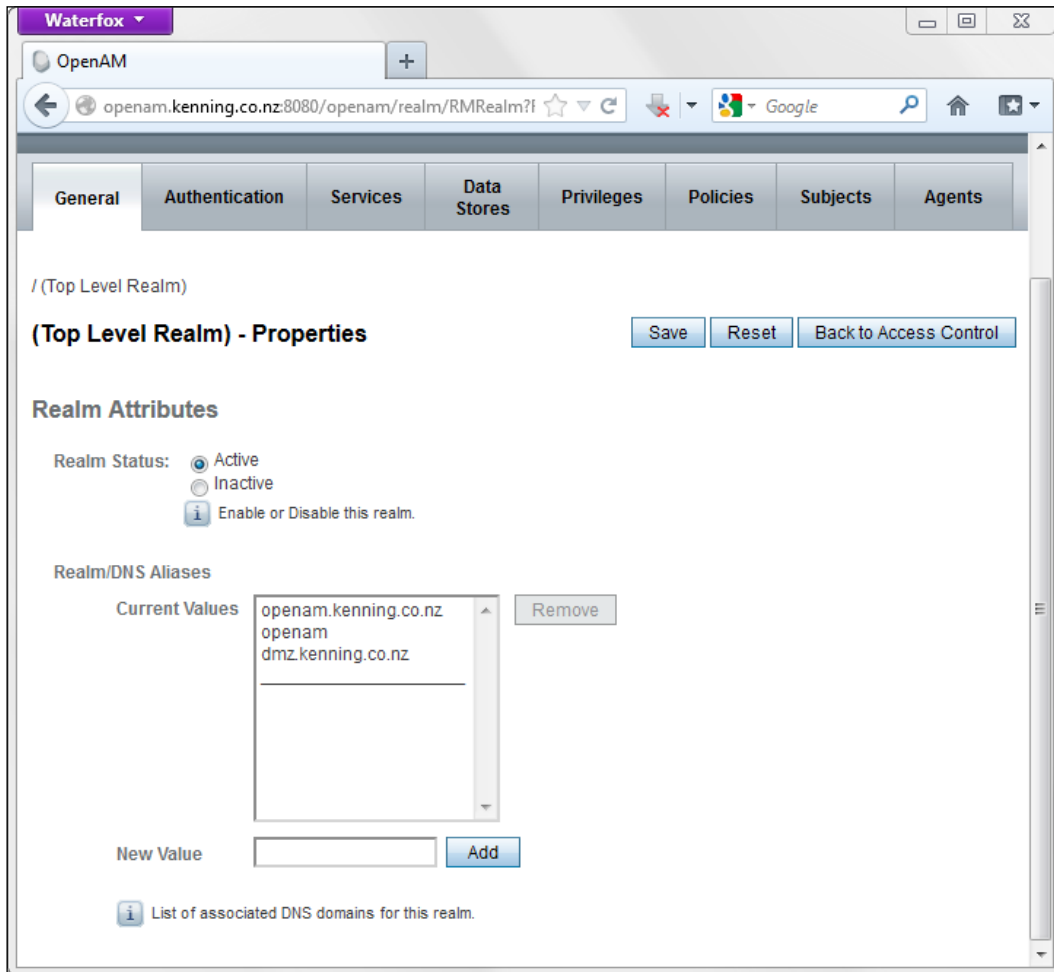


When would you use different cookie domain values?

A good example is domain names assigned to externally facing servers versus internally facing servers. Externally facing servers will need domain names valid on the Internet, such as `dmz.kenning.co.nz`. Internally facing servers may not have a valid domain name, such as `openam.kenning.local`. Hence both `.kenning.co.nz` and `.kenning.local` would need to be added as cookie domains.

4. If any changes have been made, click on the **Save** button. Next, click on the **Back to Service Configuration** button.
5. Click on the **Access Control** tab, then click on the name of your realm (mine is **/ (Top Level Realm)**).

The following screenshot shows OpenAM configured to use multiple DNS aliases for a particular realm:



6. Add the realm or DNS alias of your distributed authentication server. Mine will be `dmz.kenning.co.nz`, so enter the value in the **New Values** box, then click on the **Add** button and then on the **Save** button.

Configuring the distributed authentication application server

In this step we're going to be installing another instance of Tomcat separate from the instance running OpenAM. We'll run this instance in the DMZ with a new domain name, `dmz.kenning.co.nz`.

1. We'll be using a new domain name in this chapter. So as per the instructions in the previous chapter, open your `hosts` file and create the domain name `dmz.kenning.co.nz` and point it to `127.0.0.1`.
2. Extract the Tomcat 6 downloaded file. I extracted it to `C:\tomcat.dmz`.
3. We need to change the ports so they don't clash with the existing instance of Tomcat. Open `C:\tomcat.local\conf\server.xml` in a text editor.
4. Change all the ports from the 8000 range to the 10000 range, so the main port will go from 8080 to 10080 for example. Save the file.
5. From the command line, run `C:\tomcat.dmz\bin\startup.bat` and see if Tomcat starts.
6. Copy the file `openam-distauth-10.1.0-Xpress.war` to `C:\tomcat.dmz\webapps` folder, and rename it to `openam-distauth.war`.
7. Head to your distributed authentication application server URL, in my case this was `http://dmz.kenning.co.nz:10080/openam-distauth/`.

Configuring the distributed authentication application

Once you've executed the previous step, you should be redirected to a URL similar to `http://dmz.kenning.co.nz:10080/openam-distauth/distAuthConfigurator.jsp` which will allow you to configure your distributed authentication application.

The following screenshot shows the OpenAM distributed authentication application settings:

The screenshot shows a web browser window titled 'Configure DistAuth' with the URL 'dmz.kenning.co.nz:10080/openam-distauth/distAuthConfigurator.js'. The page content is titled 'Configuring DistAuth Application' and asks the user to 'Please provide the OpenAM Server Information.' Below this, there is a form with the following fields and values:

Server Protocol:	http
Server Host:	openam.kenning.co.nz
Server Port:	8080
Server Deployment URI:	/openam
DistAuth Server Protocol:	http
DistAuth Server Host:	dmz.kenning.co.nz
DistAuth Server Port:	10080
DistAuth Server Deployment URI:	/openam-distauth
DistAuth Cookie Name:	AMDistAuthCookie
OpenAM LB Cookie Name:	amlbcookie
DistAuth LB Cookie Name:	DistAuthLBCookieName
DistAuth LB Cookie Value:	DistAuthLBCookieValue
Debug directory	c:/tomcat.dmz/logs
Debug level	error
Encryption Key	b+NPX3Qkn2yP5rUS/d4XsBvuf
Application user name	UrlAccessAgent
Application user password	••••••••
Confirm Application user password	••••••••

At the bottom of the form are two buttons: 'Configure' and 'Reset'.

- **Server Protocol** refers to whether your OpenAM server will be using HTTPS or HTTP. For our prototype, we'll use **HTTP**.
- **Server Host** refers to the host name of your OpenAM server. Mine was `openam.kenning.co.nz`.
- **Server Port** refers to the port of your OpenAM server. Mine was `8080`.
- **Server Deployment URI** refers to where the OpenAM application was deployed to. Mine was deployed to `/openam`.



The next few fields should be pre-populated for you, and do not need to be changed.

- **DistAuth Server Protocol** refers to the protocol of the distributed authentication server. For our prototype we'll use HTTP.
- **DistAuth Server Host** refers to the host name of the distributed authentication server. Mine was `dmz.kenning.co.nz`.
- **DistAuth Server Port** refers to the port number of the distributed authentication server. Mine was `10080`.
- **DistAuth Server Deployment URI** refers to where the OpenAM distributed authentication application was deployed to. Mine was deployed to `/openam-distauth`.
- **DistAuth Cookie Name** refers to the cookie name set by the distributed authentication application. By default mine was `AMDistAuthCookie`.
- **OpenAM LB Cookie Name** refers to the cookie name set by OpenAM for load balancers. By default mine was `amlbcookie`.
- **DistAuth LB Cookie Name** refers to the cookie name set by the OpenAM distributed authentication application for load balancers. By default mine was `DistAuthLBCookieName`.
- **DistAuth LB Cookie Value** refers to the cookie value set by the OpenAM distributed authentication application for load balancers. By default mine was `DistAuthLBCookieValue`.
- **Debug directory** refers to the directory where log files will be written. On Windows these should be written with a forward slash. The location for my log files was `c:/tomcat.dmz/logs`.
- **Debug level** refers to the level of detail written to the log files. The default option is `error`, another more detailed option is `debug`. Other possible options are `message`, `warning`, `error`, or `off`.
- **Encryption Key** refers to the encryption key used to encrypt the application user password. This should not be changed.
- **Application user name** refers to the user, who should be used to authenticate this application against OpenAM. The user I used was `UrlAccessAgent`, however a more robust option would be to create a specific policy agent user for this purpose.
- **Application user password** refers to the password for the preceding user account. This password is configured during the OpenAM installation process. Mine was `agentpass`.
- Click on the **Configure** button.

You should receive the following information on screen:

DistAuth application is successfully configured.

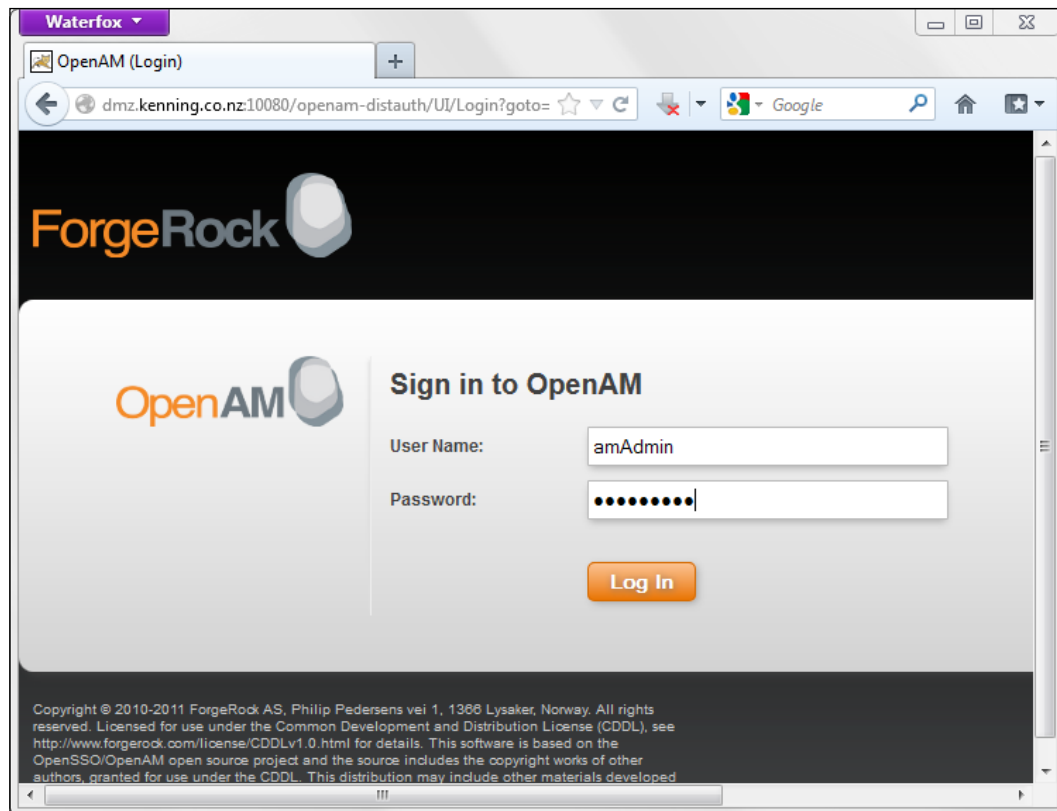
AMDistAuthConfig.properties created at C:\Users\Waylon\FAMDistAuth_tomcat.dnz_webapps_openam-distauth_AMDistAuthConfig.properties

Testing distributed authentication

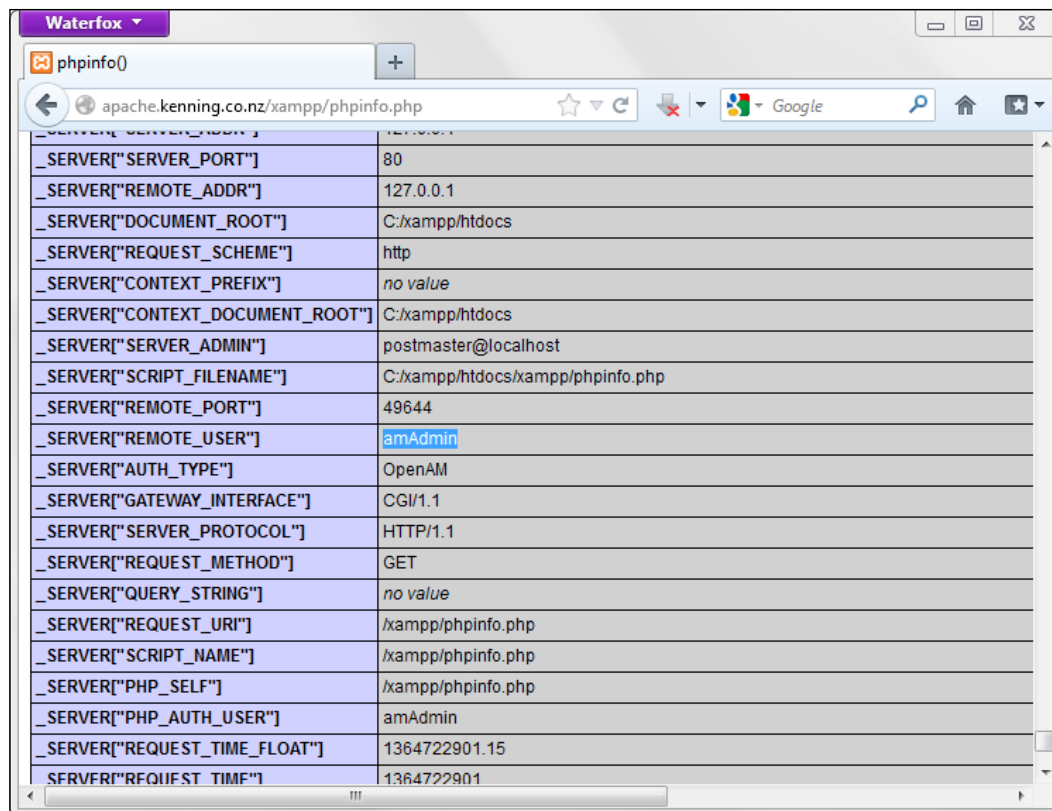
We need to create a specially crafted URL to test if distributed authentication is working successfully. This URL will look like:

PROTOCOL://DISTAUTH:PORT/DEPLOYMENT-URI/UI/

Login?goto=PROTECTEDCONTENT



A more practical example is <http://dmz.kenning.co.nz:10080/openam-distauth/UI/Login?goto=http://apache.kenning.co.nz/xampp/phpinfo.php>. This will log in a user at the distributed authentication application and then immediately (or with another redirect to `cdcservlet`, if the policy agent is running on a different domain) redirect them to a protected page on a policy agent protected web server.

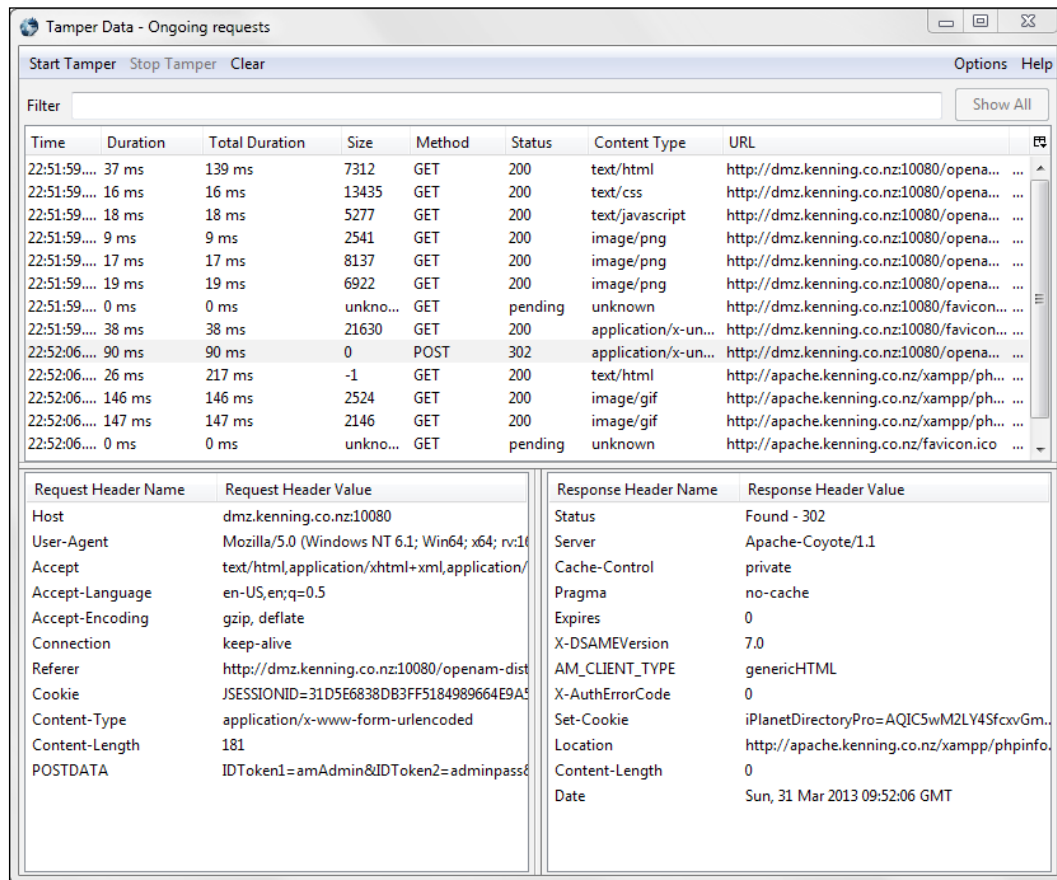


_SERVER["SERVER_NAME"]	apache.kenning.co.nz
_SERVER["SERVER_PORT"]	80
_SERVER["REMOTE_ADDR"]	127.0.0.1
_SERVER["DOCUMENT_ROOT"]	C:/xampp/htdocs
_SERVER["REQUEST_SCHEME"]	http
_SERVER["CONTEXT_PREFIX"]	no value
_SERVER["CONTEXT_DOCUMENT_ROOT"]	C:/xampp/htdocs
_SERVER["SERVER_ADMIN"]	postmaster@localhost
_SERVER["SCRIPT_FILENAME"]	C:/xampp/htdocs/xampp/phpinfo.php
_SERVER["REMOTE_PORT"]	49644
_SERVER["REMOTE_USER"]	amAdmin
_SERVER["AUTH_TYPE"]	OpenAM
_SERVER["GATEWAY_INTERFACE"]	CGI/1.1
_SERVER["SERVER_PROTOCOL"]	HTTP/1.1
_SERVER["REQUEST_METHOD"]	GET
_SERVER["QUERY_STRING"]	no value
_SERVER["REQUEST_URI"]	/xampp/phpinfo.php
_SERVER["SCRIPT_NAME"]	/xampp/phpinfo.php
_SERVER["PHP_SELF"]	/xampp/phpinfo.php
_SERVER["PHP_AUTH_USER"]	amAdmin
_SERVER["REQUEST_TIME_FLOAT"]	1364722901.15
_SERVER["REQUEST_TIME"]	1364722901

It just so happens that our protected page is a page that displays all the information received from the server such as the GET and POST variables, as well as sessions and any cookies set. As you can see in the preceding screenshot some variables are passed to the application, such as the logged in user at the variable `_SERVER["PHP_AUTH_USER"]`.

A way of verifying whether these requests went straight between the distributed authentication application and the policy agent-protected web server is to look at the HTTP requests between the two systems.

The following screenshot shows a useful Firefox extension called **Tamper Data**. One useful feature of Tamper Data is to view HTTP requests in the browser for tracing and diagnostics.



Time	Duration	Total Duration	Size	Method	Status	Content Type	URL
22:51:59....	37 ms	139 ms	7312	GET	200	text/html	http://dmz.kenning.co.nz:10080/opena...
22:51:59....	16 ms	16 ms	13435	GET	200	text/css	http://dmz.kenning.co.nz:10080/opena...
22:51:59....	18 ms	18 ms	5277	GET	200	text/javascript	http://dmz.kenning.co.nz:10080/opena...
22:51:59....	9 ms	9 ms	2541	GET	200	image/png	http://dmz.kenning.co.nz:10080/opena...
22:51:59....	17 ms	17 ms	8137	GET	200	image/png	http://dmz.kenning.co.nz:10080/opena...
22:51:59....	19 ms	19 ms	6922	GET	200	image/png	http://dmz.kenning.co.nz:10080/opena...
22:51:59....	0 ms	0 ms	unkno...	GET	pending	unknown	http://dmz.kenning.co.nz:10080/favicon...
22:51:59....	38 ms	38 ms	21630	GET	200	application/x-un...	http://dmz.kenning.co.nz:10080/favicon...
22:52:06....	90 ms	90 ms	0	POST	302	application/x-un...	http://dmz.kenning.co.nz:10080/opena...
22:52:06....	26 ms	217 ms	-1	GET	200	text/html	http://apache.kenning.co.nz/xampp/ph...
22:52:06....	146 ms	146 ms	2524	GET	200	image/gif	http://apache.kenning.co.nz/xampp/ph...
22:52:06....	147 ms	147 ms	2146	GET	200	image/gif	http://apache.kenning.co.nz/xampp/ph...
22:52:06....	0 ms	0 ms	unkno...	GET	pending	unknown	http://apache.kenning.co.nz/favicon.ico

Request Header Name	Request Header Value
Host	dmz.kenning.co.nz:10080
User-Agent	Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:10.0.1)
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language	en-US,en;q=0.5
Accept-Encoding	gzip, deflate
Connection	keep-alive
Referer	http://dmz.kenning.co.nz:10080/openam-dist
Cookie	JSESSIONID=31D5E6838DB3FF5184989664E9A5
Content-Type	application/x-www-form-urlencoded
Content-Length	181
POSTDATA	IDToken1=amAdmin&IDToken2=adminpass&

Response Header Name	Response Header Value
Status	Found - 302
Server	Apache-Coyote/1.1
Cache-Control	private
Pragma	no-cache
Expires	0
X-DSAMEVersion	7.0
AM_CLIENT_TYPE	genericHTML
X-AuthErrorCode	0
Set-Cookie	iPlanetDirectoryPro=AQIC5wM2LY4SfcxvGm...
Location	http://apache.kenning.co.nz/xampp/phpinfo...
Content-Length	0
Date	Sun, 31 Mar 2013 09:52:06 GMT

As you can see in the preceding screenshot, requests go from the distributed authentication application at `dmz.kenning.co.nz` and once authenticated, users are redirected to the value in the `GOTO` portion of the URL, in our case, `apache.kenning.co.nz`.

Summary

In this chapter we learned how to install the remote authentication application for OpenAM, and configure it to work with OpenAM. In the next chapter we'll cover application authentication with Fedlets.

5

Application Authentication with Fedlets

In the last chapter we covered distributed authentication as a way of protecting our OpenAM server. In this chapter we'll look at securing a Java application without installing a policy agent. Instead, we'll provide authentication and authorization information directly to the application using a Fedlet.

Understanding Fedlets

Fedlets are small deployable web applications that allow you to easily integrate your existing web application with OpenAM without having to write a lot of authentication and authorization code.

Advantages of Fedlets over policy agents

In our earlier chapters, we discussed policy agents, which, are installed against specific web servers and application servers, intercept requests to protected content, redirect users to OpenAM for authentication and authorization, and briefly covered how we could send that information to applications in the form of headers or session variables.

The limitation to this approach however, is the policy agent. There needs to be a specific policy agent install available for the particular web or application server you are using. If you're using a mainstream application server such as Tomcat, you're in luck. But what about if you're using something a little different, such as Nginx for serving web content? You could always install Tomcat or Apache as a reverse proxy, do authentication there with a policy agent, and then forward the requests onto your different web server, but then you've made your deployment a little more complex.

Fedlets allow you to move your authentication and authorization away from the application server layer into the application layer. As a deployable application, Fedlet sits alongside your application and looks after authentication and authorization by exchanging SAML protocol messages with OpenAM. This means no more installs against the operating system, and no more worries about whether there is a policy agent for your specific application server.

Disadvantages of Fedlets over policy agents

It's not all roses with Fedlets however. While policy agents are relatively brittle by having tight integration to a particular version of a web or application server, they do make authentication and authorization dead simple for applications. Essentially, applications shouldn't need to focus so much on authentication or authorization, rather they can focus on receiving user variables injected from the policy agent into the headers. As long as the application can read from the headers, that's all that is required.

Fedlets, being deployable web applications, instead have a reliance on what containers they can deploy to. There are two types of OpenAM Fedlets, a Java Fedlet, and a .NET Fedlet. So if you're deploying applications to Java or .NET containers, you're in luck. If you're deploying against a different language such as Ruby or PHP, you'll have to look towards third party SAML libraries to do your authentication (and turn elsewhere for authorization).

Configuring the Fedlet application server

For these instances we'll be focusing on the Java Fedlet rather than the .NET Fedlet. Once again, we're going to be installing another instance of Tomcat separate from the instance running OpenAM. We'll run this instance with a new domain name, `fedlet.kenning.co.nz`, which is not essential, but we'll use it for clarity with explanations in the chapter.

We'll be using a new domain name in this chapter, so as per the instructions in *Chapter 3, Cross-Domain Single Sign On* open your `hosts` file and create the domain name `fedlet.kenning.co.nz` and point it to `127.0.0.1`.

Extract the Tomcat 6 download file. I extracted it to `C:\tomcat.fedlet`.

We need to change the ports so they don't clash with the existing instance of Tomcat. Open `C:\tomcat.local\conf\server.xml` in a text editor.

Change all the ports from the 8000 range to the 11000 range, so the main port will go from 8080 to 11080 for instance. Save the file.

From the command line, run `C:\tomcat.fedlet\bin\startup.bat` and see if Tomcat starts.

Creating a SAML hosted identity provider

The steps for creating the identity provider are as follows:

1. Log into OpenAM. Because a Fedlet uses SAML to communicate between the application and OpenAM, we'll need to create a SAML Hosted Identity Provider.
2. Under **Common Tasks** in OpenAM, click on the **Create Hosted Identity Provider** button and you should see a screen as shown in the following screenshot:

Create a SAMLv2 Identity Provider on this Server [Configure] [Cancel]

This page allows you to configure this instance of OpenAM server as an Identity Provider (IDP). You can provide a Name for the provider, Circle of Trust (COT), its metadata of the provider and optionally Signing Certificate. A COT is a group of IDPs and Service Providers (SPs) that trust each other and in effect represents the confines within which all federation communications are performed. Metadata represents the configuration necessary to execute federation protocols (eg SAMLv2) as well as the mechanism to communicate this configuration to other entities (eg SPs) in a COT. We shall generate the metadata if you do not have one. You are required to pick a realm for this provider if there are more than one realm in the system. Otherwise, this provider will be configured under the root realm.

* Indicates required field

Do you have metadata for this provider?: ☐ Yes ☒ No ⓘ

metadata

* Name: ⓘ

Signing Key: ⓘ Please note "test" is a selfsigned certificate setup at installation for testing purposes. It is recommended you obtain a certificate from a Certificate Authority for production deployments. ⓘ

Circle of Trust

Choose from existing circles of trust listed or provide one to be created in which to include this IDP. A COT is a group of IDPs and SPs that trust each other and provides the confines within which all SAMLv2 communications are performed.

* New Circle of Trust:

Attribute Mapping

Mapping attributes helps to ensure that both the Service Provider (SP) and the Identity Provider (IDP) can recognize the same attributes that may have unique names. For example, the SP may have an attribute called UserName but the IDP may call it UserID. Eliminating these inconsistencies by mapping the attributes will guarantee that the data will be passed correctly.

Attributes Mapping	
Name in Assertion	Local Attribute Name
<input type="checkbox"/> cn	cn

[Add] ⓘ

3. For our prototype, we have no existing file that describes the metadata to be transferred between OpenAM and an application, so leave the **Do you have metadata for this provider** option on the **No** radio button.
4. In the metadata section, the first option is to enter the deployment URL of OpenAM. This will be pre-populated with your OpenAM deployment URL. Mine was `http://openam.kenning.co.nz:8080/openam`.
5. SAML communications need to be signed. This will be explained in more detail in the SAML chapter, but for now, select the **test** option under **Signing Key**.
6. In the **Circle of Trust** section, type a name for this relationship between OpenAM and the application, or circle of trust. I'll call mine 'fedlet'.
7. In the **Attribute Mapping** section we can rename variables between different systems. For instance, a variable known as `cn` for Common Name (or username) in OpenAM might be known to the application as **UserName**.
8. We need to enter values here which will be transferred between OpenAM and the Fedlet. Enter `cn` in the textbox on the left, `cn` in the textbox on the right, and select **cn** from the dropdown and click on the **Add** button.
9. Click on the **Configure** button.
10. A confirmation page will appear stating **Your Identity Provider has been configured**. On this page click on the link to create a Fedlet.

Creating a Fedlet

The steps for creating a Fedlet are as follows:

1. Because we're creating the Fedlet immediately after creating a Hosted Identity Provider, selecting the **Circle of Trust** and **Identity Provider** options have already been pre-populated as shown in the following screenshot.

Firefox

OpenAM

openam.kenning.co.nz:8080/openam/task/CreateFedlet

VERSION LOG OUT

User: amAdmin Server: Waylon-PC

Create Fedlet

Create Cancel

The Fedlet is ideal for an IDP that needs to enable an SP that does not have any kind of federation solution in place. A Fedlet is a very small zip file that you can provide a service provider (SP) so they can instantaneously federate with you. The SP simply adds the Fedlet to their application, deploys their application and they are federation enabled.

* Indicates required field

* Circle of Trust: fedlet

* Identity Provider: http://openam.kenning.co.nz:8080/openam

Fedlet information

* Name: http://fedlet.kenning.co.nz:11080/fedlet

* Destination URL of the Service Provider which will include the Fedlet: http://fedlet.kenning.co.nz:11080/fedlet

Attribute Mapping

Mapping attributes helps to ensure that both the Service Provider (SP) and the Identity Provider (IDP) can recognize the same attributes that may have unique names. For example, the SP may have an attribute called UserName but the IDP may call it UserID. Eliminating these inconsistencies by mapping the attributes will guarantee that the data will be passed correctly.

Attributes Mapping

Delete

Name in Assertion	Local Attribute Name
cn	cn

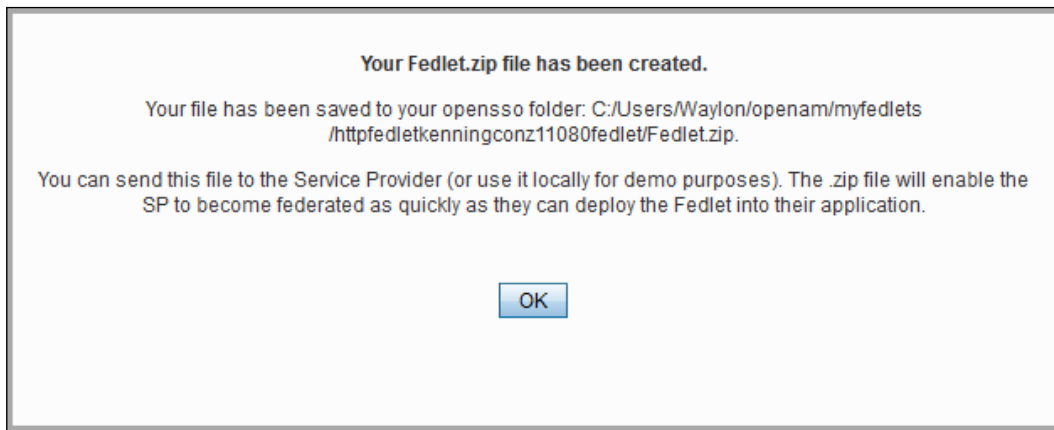
Add

cn

- Under Fedlet information, enter the name and destination URL where the Fedlet will be located. These can both have the same value, which in my case will be `http://fedlet.kenning.co.nz:11080/fedlet`.
- Once again, in the **Attribute Mapping** section we can rename variables between the different systems. For instance, a variable known as `cn` for Common Name (or username) in OpenAM might be known to the application as **UserName**.

4. We still need to enter values here which will be transferred between OpenAM and the Fedlet. Enter `cn` in the text box on the left, `cn` in the textbox on the right, and select `cn` from the dropdown and click on the **Add** button.
5. Click on the **Create** button.

Inside the browser a message will appear saying your `Fedlet.zip` has been created and saved to an `opensso` folder as shown in the following screenshot. For me this location was `C:/Users/Waylon/openam/myfedlets/httpfedletkenningconz11080fedlet/Fedlet.zip`.



Deploying Fedlet.zip onto our Java application server

Extract from `Fedlet.zip`, the `fedlet.war` file. Copy `fedlet.war` to `C:\tomcat.fedlet\webapps`. Tomcat will automatically start deploying the web application `Fedlet`.

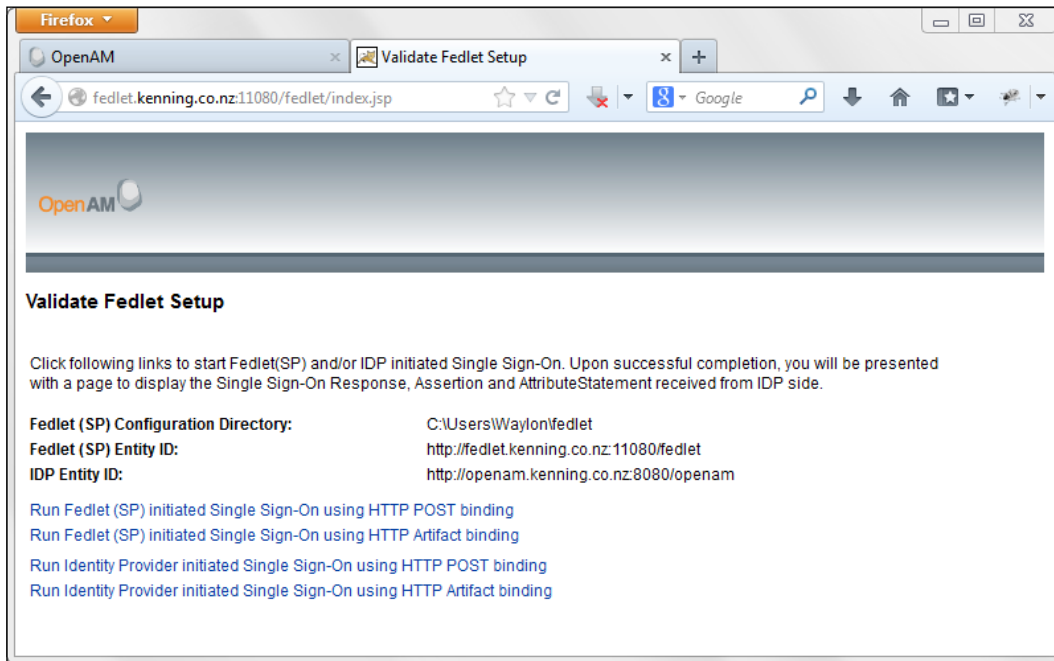
In a browser, go to the Fedlet deployment URL. For me, this is `http://fedlet.kenning.co.nz:11080/fedlet/`.

This page will state that the Fedlet configuration home directory does not exist and provides you with a link to create this directory. For me, this link was `http://fedlet.kenning.co.nz:11080/fedlet/index.jsp?CreateConfig=true`.

Clicking on the link will return a message that a Fedlet configuration has been created in a folder, which was `C:\Users\Waylon\fedlet` (for my application). Click on the link on the page, which was `http://fedlet.kenning.co.nz:11080/fedlet/index.jsp` (for my application), to continue.

Validating the Fedlet setup

The following page allows us to validate that our Fedlet can communicate and authenticate against our OpenAM server.



There are four links:

- Fedlet initiated SSO using HTTP POST binding
- Fedlet initiated SSO using HTTP Artifact binding
- Identity Provider (OpenAM) initiated SSO using HTTP POST binding
- Identity Provider (OpenAM) initiated SSO using HTTP Artifact binding

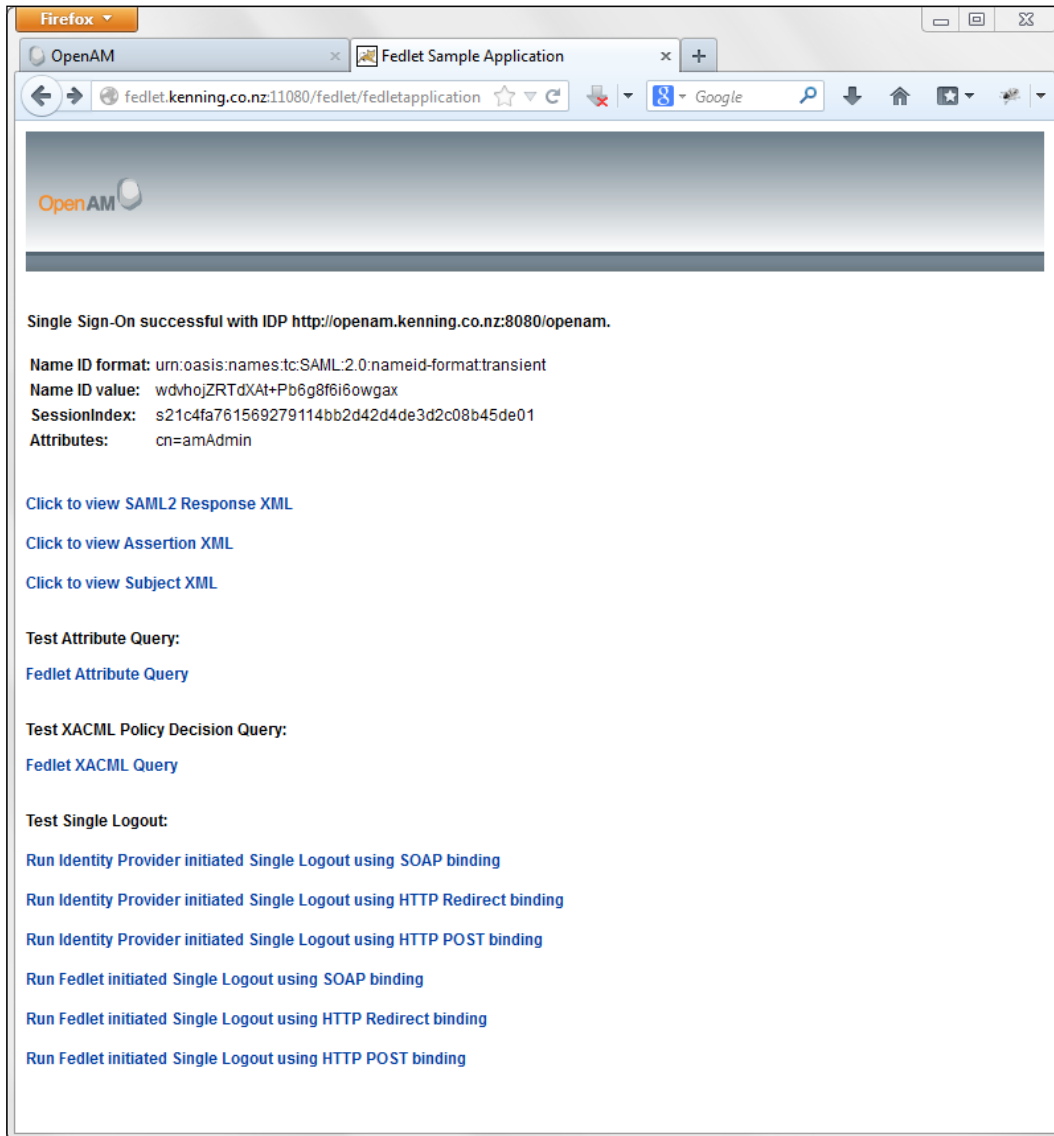
There are two key differences in these links. The first is whether SSO is initiated from the Fedlet, or OpenAM.

If SSO is initiated from the Fedlet, the request goes to the Fedlet first, which redirects it to OpenAM, which then redirects it back to the Fedlet.

If SSO is initiated from OpenAM, the request goes to OpenAM first, which then redirects it to the Fedlet.

The second key difference is around the type of binding used to communicate the SAML messages. These can either be via an HTTP POST request using a hidden field, or by locating a SAML response at a particular URL known as an HTTP Artifact.

Clicking on these links should return a confirmation page saying Single Sign On successful as shown in the following screenshot:



You can see that the attribute we wanted, **cn**, is listed as an attribute. **cn** happens to be the username of the currently logged in user. You can click on any of the links to view various XML used in the transaction in more detail.

On this page you can also test Single Logout using either Fedlet initiated SSO, or OpenAM initiated SSO, using a variety of bindings.

More information about Fedlets

The OpenAM documentation about Fedlets contains a lot of great detail about customizing the Fedlet package into your Java application. This documentation can be found at <http://openam.forgerock.org/openam-documentation/openam-doc-source/doc/dev-guide/index/chap-fedlet-java.html>.

Summary

In this chapter we created another Tomcat installation, created a Hosted Identity Provider, created a Fedlet which trusts our Hosted Identity Provider, deployed it to our new Tomcat installation, configured the Fedlet, and tested Single Sign On and Single Sign Out.

6

Implementing SAML2 Federation Patterns

In the last chapter we covered setting up authentication at the application level using Fedlets. Fedlets rely on SAML as their communication method to OpenAM. This chapter will cover:

- Understanding SAML
- Configuring SAML in OpenAM
- Testing our SAML connection between OpenAM and a PHP-enabled SAML application

Understanding SAML

SAML stands for **Security Assertion Markup Language** which is a protocol used to exchange authentication and authorization messages between two parties using an XML format.

The two types of parties in SAML are called **Service Providers (SP)** and **Identity Providers (IdP)**. These two parties are aware of each other and have a relationship known in OpenAM as a **Circle of Trust**.

Understanding Identity Providers

An Identity Provider is the party that provides identity to a user within a security realm. Google Accounts would be an example of an Identity Provider.

In our example, OpenAM will act as an Identity Provider, providing the identities stored within the directory (which for us are the identities contained within the embedded OpenDJ instance).

Understanding Service Providers

A Service Provider provides a service to a user. Google Plus or Gmail are Service Providers which rely on an Identity Provider, Google Accounts, to provide identity information to the Service Provider.

Understanding a Circle of Trust

Not any Identity Provider is trusted by any Service Provider. I can't sign into Gmail using my Facebook username and password, and my bank doesn't trust my work username and password. So we know that there needs to be a relationship between the Service Provider and the Identity Provider, which will be formally defined as a Circle of Trust.

Configuring OpenAM as a SAML Identity Provider

The steps for configuring OpenAM are as follows:

1. Sign into the OpenAM console.
2. Under the **Common Tasks** tab, click on the **Create Hosted Identity Provider** button and you will see a screen as shown in the following screenshot:

The screenshot shows a web browser window with the OpenAM console. The page title is "Create a SAML2 Identity Provider on this server". The URL is "openam.kenning.co.nz:8080/openam/task/CreateHosted". The page contains a form for configuring a new SAML2 Identity Provider. The form has a "metadata" section and a "Circle of Trust" section. The "metadata" section includes a "Name" field (required) with the value "http://openam.kenning.co.nz:8080/openam", a "Signing Key" dropdown menu set to "test", and a note: "Please note 'test' is a selfsigned certificate setup at installation for testing purposes. It is recommended you obtain a certificate from a Certificate Authority for production deployments." The "Circle of Trust" section includes a "Circles of Trust" section with radio buttons for "Add to existing" and "Add to new", and a "New Circle of Trust" field with the value "simpleSAMLphp".

Firefox

OpenAM

openam.kenning.co.nz:8080/openam/task/CreateHosted

Create a SAML2 Identity Provider on this server

This page allows you to configure this instance of OpenAM server as an Identity Provider (IDP). You can provide a Name for the provider, Circle of Trust (COT), its metadata of the provider and optionally Signing Certificate. A COT is a group of IDPs and Service Providers (SPs) that trust each other and in effect represents the confines within which all federation communications are performed. Metadata represents the configuration necessary to execute federation protocols (eg SAML2) as well as the mechanism to communicate this configuration to other entities (eg SPs) in a COT. We shall generate the metadata if you do not have one. You are required to pick a realm for this provider if there are more than one realm in the system. Otherwise, this provider will be configured under the root realm.

* Indicates required field

Do you have metadata for this provider?: ☐ Yes ☒ No

metadata

* Name:

Signing Key: Please note "test" is a selfsigned certificate setup at installation for testing purposes. It is recommended you obtain a certificate from a Certificate Authority for production deployments.

Circle of Trust

Choose from existing circles of trust listed or provide one to be created in which to include this IDP. A COT is a group of IDPs and SPs that trust each other and provides the confines within which all SAML2 communications are performed.

Circles of Trust: ☐ Add to existing ☒ Add to new

* New Circle of Trust:

3. For advanced configurations of OpenAM as a SAML Identity Provider we may already have metadata about this Identity Provider pre-configured, which can be loaded into OpenAM. However, for our install we'll leave the option **Do you have metadata for this provider?** as **No**.
4. Under the metadata section we need to define the name of the Identity Provider. This should be your OpenAM deployment URL. Mine was `http://openam.kenning.co.nz:8080/openam`.



One thing to note in our prototype is we're using HTTP rather than HTTPS connections. Generally it is always preferable to use HTTPS connections. However, one of the features of SAML 2.0 is the introduction of encryption of SAML. Even though we're serving our SAML content over an unencrypted HTTP connection, we've encrypted the SAML message itself.

5. We need to select a **Signing Key** to sign our certificate against. By default, there is only one key available, that is, **test**. This is a self-signed certificate and isn't suitable for a production environment, however, for our prototype it'll do just fine.



Understanding certificates and Java

Certificates used for encryption are generally found inside a **Java Key Store (JKS)**. The creation of a certificate is beyond the scope of this book but can be found at <http://docs.oracle.com/javase/1.5.0/docs/tooldocs/solaris/keytool.html>.

Once a certificate is created, it needs to be imported into the JKS used by OpenAM, located in the installation directory of OpenAM. **Portecle** is a GUI-based open source application that can easily do this, and is found at <http://portecle.sourceforge.net/>.

- Next, we need to create a **Circle of Trust**. This Circle of Trust defines the relationship between an Identity Provider (OpenAM), and a Service Provider (SimpleSAMLphp for our prototype). I named my Circle of Trust SimpleSAMLphp.

Firefox

OpenAM

openam.kenning.co.nz:8080/openam/task/CreateHosted

New Circle of Trust: simpleSAMLphp

Attribute Mapping

Mapping attributes helps to ensure that both the Service Provider (SP) and the Identity Provider (IDP) can recognize the same attributes that may have unique names. For example, the SP may have an attribute called UserName but the IDP may call it UserID. Eliminating these inconsistencies by mapping the attributes will guarantee that the data will be passed correctly.

Attributes Mapping

Delete

	Name in Assertion	Local Attribute Name
<input type="checkbox"/>	cn	cn
<input type="checkbox"/>	employeeenumber	employeeenumber
<input type="checkbox"/>	email	mail
<input type="checkbox"/>	manager	manager
<input type="checkbox"/>	address	postaladdress
<input type="checkbox"/>	surname	sn
<input type="checkbox"/>	givenname	givenname
<input type="checkbox"/>	phone	telephonenumber


uid uid Add

uid


- Next, we need to map the attributes. These are the values that we pass between OpenAM and our third-party system. Select from the attributes available in the drop-down box, which will populate the textbox on the right. Type the name of the attribute you would like. For example, I had **sn** selected on the right, but gave the attribute the name of **surname** instead.
- Click on the **Configure** button at the top of the page.

Installing SimpleSAMLphp


We need to integrate a SAML Service Provider with OpenAM acting as a SAML Identity Provider. For Java or .NET applications, we could use Fedlets like in the last chapter. But let's use a non-OpenAM technology, SimpleSAMLphp.

 SimpleSAMLphp is a lightweight SAML Identity Provider and Service Provider written in PHP, and can be deployed to any web server, such as Apache.

1. Download the latest version of SimpleSAMLphp from <http://simplesamlphp.org/download>. The version I'm using is 1.10.0.

 If you're going to use the web server we installed the Apache Policy Agent on earlier, don't forget to uninstall the Apache Policy Agent.

2. Since I'm going to use the same Apache web server as used earlier in the book, I'm extracting the contents of the archive to `c:\xampp\simplesaml`.

 There is a problem with XAMPP 1.8 and OpenSSL. At the time of writing, XAMPP 1.7 works correctly.

3. Add the following contents to your `httpd-vhosts.conf` file, which for me was located at the path: `C:\xampp\apache\conf\extra\httpd-vhosts.conf`:

```
<VirtualHost apache.kenning.co.nz:80>
    DocumentRoot "C:/xampp/simplesaml"
    ServerName apache.kenning.co.nz
    ErrorLog "logs/simplesaml.local-error.log"
    CustomLog "logs/simplesaml.local-access.log" combined
    Alias /simplesaml "c:/xampp/simplesaml/www"
    ServerAlias apache.kenning.co.nz
    <Directory "C:/xampp/simplesaml">
        AllowOverride All
        Order Allow,Deny
        Allow from all
        Require all granted
    </Directory>
</VirtualHost>
```


4. Restart your Apache Web Server for these settings to take effect. These settings allow us to access SimpleSAMLphp by the URL `http://apache.kenning.co.nz/simplesaml/`.
5. PHP OpenSSL support is required for SimpleSAMLphp. If you're using XAMPP like me, open the file `C:\xampp\php\php.ini` and remove the semicolon on the line `extension=php_openssl.dll`.
6. Edit the file `C:\xampp\simplesaml\config\config.php` and change the line containing `auth.adminpassword` from `123` to `password`. Also change the line referring to `secretsalt` from `defaultsecretsalt` to `secretsalt`.
7. Inside the folder `C:\xampp\simplesaml\modules` create a new blank file called `enable`. This will enable the modules we need to set up SimpleSAMLphp as a Service Provider.
8. If things are going well, accessing `http://apache.kenning.co.nz/simplesaml/module.php/core/frontpage_config.php` should show all required modules with a green tick. If not, read the SimpleSAMLphp install notes located at `http://simplesamlphp.org/docs/stable/simplesamlphp-install`.

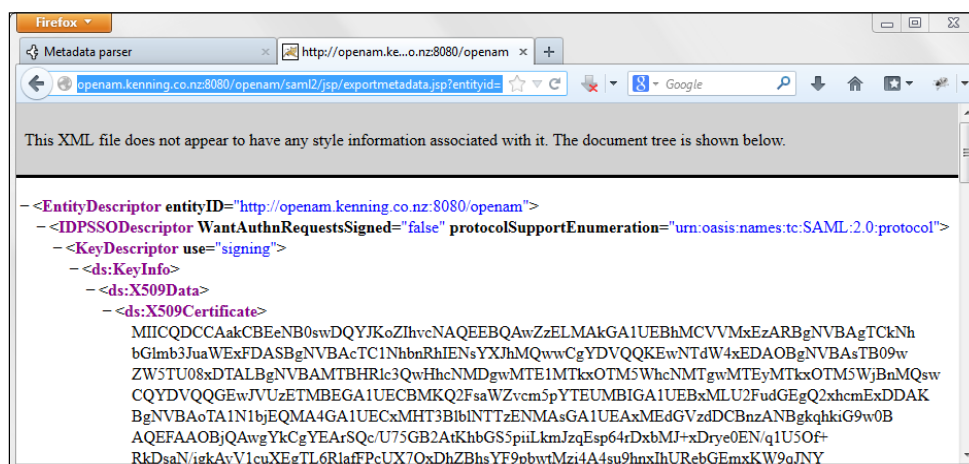
Configuring SimpleSAMLphp as a Service Provider

Now we need to add the metadata about our OpenAM IdP to SimpleSAMLphp. But first, we need to convert the OpenAM IdP metadata into a format usable by SimpleSAMLphp.

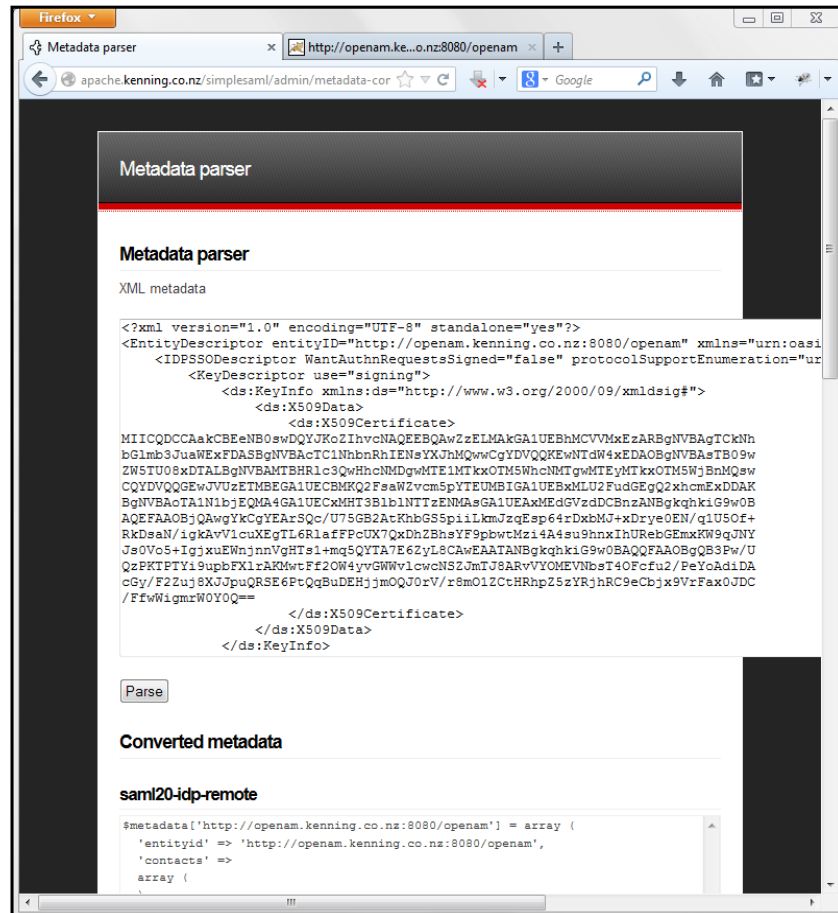
1. Inside OpenAM, click on the **Federation** tab at the top, and note the name of your Entity Provider as shown in the following screenshot. This will be used as the value at the end of the string in the next URL.

Entity Providers (1 Item(s))					
<input type="button" value="New..."/> <input type="button" value="Delete"/> <input type="button" value="Import Entity..."/>					
<input checked="" type="checkbox"/>	Name	Protocol	Type	Location	Realm
<input type="checkbox"/>	http://openam.kenning.co.nz:8080/openam	SAMLv2	IDP	Hosted	/

2. We now need to provide OpenAM's IdP Metadata to SimpleSAMLphp so the two systems know to trust each other. Head to `http://openam.kenning.co.nz:8080/openam/saml2/jsp/exportmetadata.jsp?entityid=http://openam.kenning.co.nz:8080/openam` which should return an XML result similar to the following screenshot:



3. Copy that XML and paste into <http://apache.kenning.co.nz/simplesaml/admin/metadata-converter.php> then click on **Parse**. You should then be returned with a formatted list of metadata as shown in the following screenshot:



4. Copy that formatted metadata and paste into the bottom of the file `c:\xampp\simplesaml\metadata\saml20-idp-remote.php` and save the file. This lets SimpleSAMLphp know about our OpenAM IdP.

Configuring OpenAM to trust a SimpleSAMLphp SP

Next we need to take our SimpleSAMLphp SP metadata and provide that back to OpenAM. After this, SimpleSAMLphp will be aware of OpenAM as a valid IdP, and OpenAM will be aware of SimpleSAMLphp as a valid SP.

1. Inside SimpleSAMLphp click on the **Federation** tab, and then click on **Show metadata**. This will redirect you to the URL `http://apache.kenning.co.nz/simplesaml/module.php/saml/sp/metadata.php/default-sp?output=xhtml` which lists the metadata for our SimpleSAMLphp SP. We're also provided all that metadata in a URL `http://apache.kenning.co.nz/simplesaml/module.php/saml/sp/metadata.php/default-sp` so we should copy this URL.
2. In the OpenAM console under **Common Tasks**, click on the **Register Remote Service Provider** button. You will then see a screen as shown in the following screenshot:

Create a SAML2 Remote Service Provider Configure Cancel

This page allows you to register a remote Service Provider (SP). You need two things: Circle of Trust (COT) and metadata of the provider. A COT is a group of Identity Providers (IDPs) and SPs that trust each other and in effect represents the confines within which all federation communications are performed. Metadata represents the configuration necessary to execute federation protocols (eg SAMLv2) as well as the mechanism to communicate this configuration to other entities (eg IDPs) in a COT.

Where does the metadata file reside?: ☒ URL ☐ File * Indicates required field

* URL where metadata is located:

Circle of Trust

Choose from existing circles of trust listed or provide one to be created in which to include this SP. A COT is a group of IDPs and SPs that trust each other and provides the confines within which all SAMLv2 communications are performed.

Circles of Trust: ☒ Add to existing ☐ Add to new

* Existing Circle of Trust:

Attribute Mapping

Name in Assertion	Local Attribute Name
<input type="checkbox"/> cn	cn
<input type="checkbox"/> employeeenumber	employeeenumber
<input type="checkbox"/> email	mail
<input type="checkbox"/> manager	manager
<input type="checkbox"/> address	postaladdress
<input type="checkbox"/> surname	sn
<input type="checkbox"/> givenname	givenname
<input type="checkbox"/> phone	telephonenumber

Delete Add

3. Copy the metadata link into the URL where metadata is located, then click on the **Configure** button. You will then see a screen as shown in the following screenshot:

The screenshot displays two sections of a web interface. The top section, titled 'Circle of Trust (1 Item(s))', contains a table with one entry for 'simpleSAMLphp'. The bottom section, titled 'Entity Providers (2 Item(s))', contains a table with two entries: a Service Provider (SP) and an Identity Provider (IDP).

Circle of Trust (1 Item(s))				
<input type="button" value="New..."/> <input type="button" value="Delete"/>				
<input checked="" type="checkbox"/>	Name	Entities	Realm	Status
<input type="checkbox"/>	simpleSAMLphp	http://apache.kenning.co.nz/simplesaml/module.php/saml/sp/metadata.php/default-sp saml2 http://openam.kenning.co.nz:8080/openam saml2	/	Active

Entity Providers (2 Item(s))					
<input type="button" value="New..."/> <input type="button" value="Delete"/> <input <="" th="" type="button" value="Import Entity..."/>					
<input checked="" type="checkbox"/>	Name	Protocol	Type	Location	Realm
<input type="checkbox"/>	http://apache.kenning.co.nz/simplesaml/module.php/saml/sp/metadata.php/default-sp	SAMLv2	SP	Remote	/
<input type="checkbox"/>	http://openam.kenning.co.nz:8080/openam	SAMLv2	IDP	Hosted	/

Our Circle of Trust will show both the OpenAM IdP and the SimpleSAMLphp SP as shown in the preceding screenshot.

Testing our SAML Circle of Trust

The steps for testing SAML are as follows:

1. Head to http://apache.kenning.co.nz/simplesaml/module.php/core/frontpage_welcome.php and click on the **Authentication** tab, and then the **Test configured authentication sources** link.
2. On the **Test authentication sources** page, click on the **default-sp** link.
3. On the **Select your identity provider** page, select the OpenAM IdP we configured and click on the **Select** button.

This will then show all the SAML identity attributes passed from OpenAM to SimpleSAMLphp.

Summary

In this chapter we talked about the terminology of SAML, created a SAML IdP in OpenAM, created a SAML SP using SimpleSAMLphp, and configured them to work together in an OpenAM Circle of Trust. In the next chapter we'll be covering OAuth authentication.

7

OAuth Authentication

In the last chapter we covered SAML which is used to communicate identity information between Identity Providers and Service Providers. In that instance we used OpenAM as the Provider of the identity information. But that won't always be the case, we might want to allow authentication against Facebook or Google. This chapter will explain how to:

- Use Facebook as OAuth provider
- Configure OAuth module
- Configure authenticating chaining
- Test OAuth Client

Understanding OAuth

OAuth has the concept of **Providers** and **Clients**. An OAuth Provider is like a SAML Identity Provider, and is the place where the user enters their authentication credentials. Typical OAuth Providers include Facebook and Google.

OAuth Clients are resources that want to protect resources, such as a SAML Service Provider. If you have ever been to a site that has asked you to log in using your Twitter or LinkedIn credentials then odds are that site was using OAuth.

The advantage of OAuth is that a user's authentication credentials (username and password, for instance) is never passed to the OAuth Client, just a range of tokens that the Client requested from the Provider and which are authorized by the user.

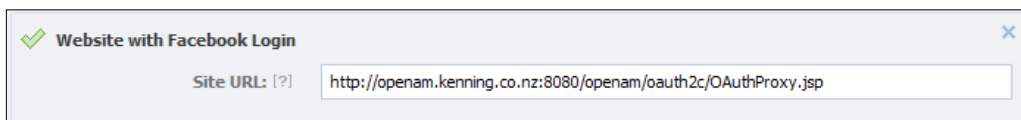
OpenAM can act as both an OAuth Provider and an OAuth Client. This chapter will focus on using OpenAM as an OAuth Client and using Facebook as an OAuth Provider.

Preparing Facebook as an OAuth Provider

Head to <https://developers.facebook.com/apps/> and create a **Facebook App**. Once this is created, your Facebook App will have an **App ID** and an **App Secret**. We'll use these later on when configuring OpenAM.

Facebook won't let a redirect to a URL (such as our OpenAM installation) without being aware of the URL. The steps for preparing Facebook as an OAuth provider are as follows:

1. Under the settings for the App in the section **Website with Facebook Login** we need to add a **Site URL**. This is a special OpenAM OAuth Proxy URL, which for me was `http://openam.kenning.co.nz:8080/openam/oauth2c/OAuthProxy.jsp` as shown in the following screenshot:



2. Click on the **Save Changes** button on Facebook.



My OpenAM installation for this chapter was directly available on the Internet just in case Facebook checked for a valid URL destination.

Configuring an OAuth authentication module

OpenAM has the concept of authentication modules, which support different ways of authentication, such as OAuth, or against its Data Store, or LDAP or a Web Service. We need to create a new Module Instance for our Facebook OAuth Client.

1. Log in to OpenAM console. Click on the **Access Control** tab, and click on the link to the realm/ (**Top Level Realm**).
2. Click on the **Authentication** tab and scroll down to the **Module Instances** section. Click on the **New** button.

3. Enter a name for the **New Module Instance** and select **OAuth 2.0** as the **Type** and click on the **OK** button. I used the name `Facebook`. You will then see a screen as shown:

The screenshot shows the OpenAM web interface in a Firefox browser. The address bar shows the URL `openam.kenning.co.nz:8080/openam/authentication`. The page title is "OpenAM". The user is logged in as "amAdmin" on "Server: Waylon-PC". The "LOG OUT" button is visible in the top right corner.

The main content area is titled "OAuth 2.0" and includes "Save", "Reset", and "Back to Authentication" buttons. Below this is the "Realm Attributes" section, which contains the following fields:

- Client Id:** `187529064617026` (with a help icon and "OAuth client_id parameter" text)
- Client Secret:** (masked with dots) (with a help icon and "OAuth client_secret parameter" text)
- Client Secret (confirm):** (masked with dots)
- Authentication Endpoint URL:** `https://www.facebook.com/dialog/oauth` (with a help icon and "OAuth authentication endpoint URL" text)
- Access Token Endpoint URL:** `https://graph.facebook.com/oauth/acces` (with a help icon and "OAuth access token endpoint URL" text)
- User Profile Service URL:** `https://graph.facebook.com/me` (with a help icon and "User profile information URL" text)
- Scope:** `email_read_stream` (with a help icon and "OAuth scope; list of user profile properties" text)
- Proxy URL:** `http://openam.kenning.co.nz:8080/openam` (with a help icon and "The URL to the OpenAM OAuth proxy JSP" text)
- Account Mapper:** `org.forgerock.openam.authentication.mc` (with a help icon and "Name of the class implementing the account mapping" text)

4. For **Client Id**, use the **App ID** value provided from Facebook. For the **Client Secret** use the **App Secret** value provided from Facebook as shown in the preceding screenshot.
5. Since we're using Facebook as our OAuth Provider, we can leave the **Authentication Endpoint URL**, **Access Token Endpoint URL**, and **User Profile Service URL** values as their default values.

6. **Scope** defines the permissions we're requesting from the OAuth Provider on behalf of the user. These values will be provided by the OAuth Provider, but we'll use the default values of **email** and **read_stream** as shown in the preceding screenshot.
7. **Proxy URL** is the URL we copied to Facebook as the **Site URL**. This needs to be replaced with your OpenAM installation value.

The screenshot shows the OpenAM web interface in a Firefox browser window. The address bar shows the URL `openam.kenning.co.nz:8080/openam/authentication`. The page title is "OpenAM".

Account Mapper Configuration

Current Values: `email=mail`
`id=facebook-id` [Remove]

New Value: [] [Add]

i Mapping of OAuth account to local OpenAM account

Attribute Mapper: `org.forgerock.openam.authentication.mc`
i Name of the class that implements the attribute mapping

Attribute Mapper Configuration

Current Values: `last_name=sn`
`email=mail`
`last_name=facebook-lname`
`first_name=givenname`
`first_name=facebook-fname`
`id=facebook-id`
`email=facebook-email`
`name=cn` [Remove]

New Value: [] [Add]

i Mapping of OAuth attributes to local OpenAM attributes

8. The **Account Mapper Configuration** allows you to map values from your OAuth Provider to values that OpenAM recognizes. For instance, Facebook calls emails **email** while OpenAM references values from the directory it is connected to, such as **mail** in the case of the embedded LDAP server. This goes the same for the **Attribute Mapper Configuration**. We'll leave all these sections as their defaults as shown in the preceding screenshot.

The screenshot shows the OpenAM configuration interface in a Firefox browser window. The address bar shows the URL: `openam.kenning.co.nz:8080/openam/authentica`. The page contains several configuration sections:

- Save attributes in the session:** ☒ Enabled. Description: If this option is enabled, the attributes configured in the attribute mapper will be saved into the OpenAM session.
- Email attribute in OAuth2 Response:** . Description: Attribute from the OAuth2 response used to send activation code emails.
- Create account if it does not exist:** ☒ Enabled. Description: If the OAuth2 account does not exist in the local OpenAM data store, an account will be created dynamically.
- Prompt for password setting and activation code:** ☐ Enabled. Description: Users must set a password and complete the activation flow during dynamic profile creation.
- Map to anonymous user:** ☐ Enabled. Description: Enabled anonymous user access to OpenAM for OAuth authenticated users.
- Anonymous User:** anonymous. Description: Username of the OpenAM anonymous user.
- OAuth 2.0 Provider logout service:** . Description: The URL of the OAuth Identity Providers Logout service.
- Logout options:**
 - ☐ Do not logout
 - ☐ Log out
 - ☒ Prompt
Description: Controls how Logout options will be presented to the user.
- Mail Server Gateway implementation class:** org.forgerock.openam.authentication.mc. Description: The class used by the module to send email.
- SMTP host:** localhost. Description: The mail host that will be used by the Email Gateway implementation.
- SMTP port:** 25. Description: The TCP port that will be used by the SMTP gateway.
- SMTP User Name:** . Description: If the SMTP Service requires authentication, configure the user name here.
- SMTP User Password:** . Description: The Password of the SMTP User Name.
- SMTP User Password (confirm):** .
- SMTP SSL Enabled:** ☐ Enabled.

- OpenAM allows attributes passed from the OAuth Provider to be saved to the OpenAM session. We'll make sure this option is **Enabled** as shown in the preceding screenshot.

10. When a user authenticates against an OAuth Provider, they are likely to not already have an account with OpenAM. If they do not have a valid OpenAM account then they will not be allowed access to resources protected by OpenAM. We should make sure that the option to **Create account if it does not exist** is **Enabled** as shown in the preceding screenshot.



Forcing authentication against particular authentication modules

In the writing of this book I disabled the **Create account if it does not exist** option while I was testing. Then when I tried to log into OpenAM I was redirected to Facebook, which then passed my credentials to OpenAM. Since there was no valid OpenAM account that matched my Facebook credentials I could not log in. For your own testing, it would be recommended to use `http://openam.kenning.co.nz:8080/openam/UI/Login?module=Facebook` rather than changing your authentication chain.

Thankfully, you can force a login using a particular authentication module by adjusting the login URL. By using `http://openam.kenning.co.nz:8080/openam/UI/Login?module=DataStore`, I was able to use the Data Store rather than OAuth authentication module, and log in successfully.

11. For our newly created accounts we can choose to prompt the user to create a password and enter an activation code. For our prototype we'll leave this option as **Disabled**.

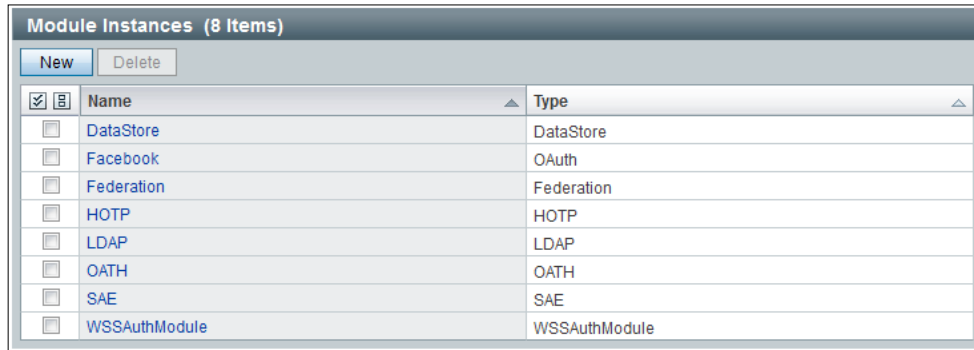
The flip side to Single Sign On is Single Log Out. Your OAuth Provider should provide a logout URL which we could possibly call to log out a user when they log out of OpenAM. The options we have when a user logs out of OpenAM is to either not log them out of the OAuth Provider, to log them out of the OAuth Provider, or to ask the user.

If we had set earlier that we wanted to enforce password and activation token policies, then we would need to enter details of an SMTP server, which would be used to email the activation token to the user. For the purposes of our prototype we'll leave all these options blank.

12. Click on the **Save** button.

Configuring Authentication Chaining

In the following screenshot we show all the authentication modules available for us to use:



The screenshot shows a window titled "Module Instances (8 Items)". It has a "New" button and a "Delete" button. Below these is a table with two columns: "Name" and "Type". The table contains eight rows of data.

<input checked="" type="checkbox"/> <input type="checkbox"/>	Name	Type
<input type="checkbox"/>	DataStore	DataStore
<input type="checkbox"/>	Facebook	OAuth
<input type="checkbox"/>	Federation	Federation
<input type="checkbox"/>	HOTP	HOTP
<input type="checkbox"/>	LDAP	LDAP
<input type="checkbox"/>	OATH	OATH
<input type="checkbox"/>	SAE	SAE
<input type="checkbox"/>	WSSAuthModule	WSSAuthModule

You'll notice that we have a lot of authentication modules. We don't have to use just one authentication module, we can have multiple options. But then how do we determine which ones we want to use, and how important are they to us?

We do this in the section Authentication Chaining, where we chain authentication modules together. For our prototype we're going to change the default authentication chain, but in a production environment it would be recommended to create a new authentication chain.

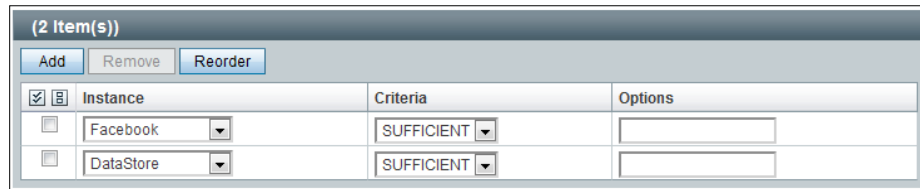
1. In the **Authentication Chaining** section click on the **IdapService** link as shown in the following screenshot:



The screenshot shows a window titled "Authentication Chaining (1 Items)". It has a "New" button and a "Delete" button. Below these is a table with one column: "Name". The table contains one row of data.

<input checked="" type="checkbox"/> <input type="checkbox"/>	Name
<input type="checkbox"/>	IdapService

2. Click on the **Add** button to add another Authentication Instance. Select the OAuth authentication module we created earlier. Mine is named **Facebook** as shown in the following screenshot:



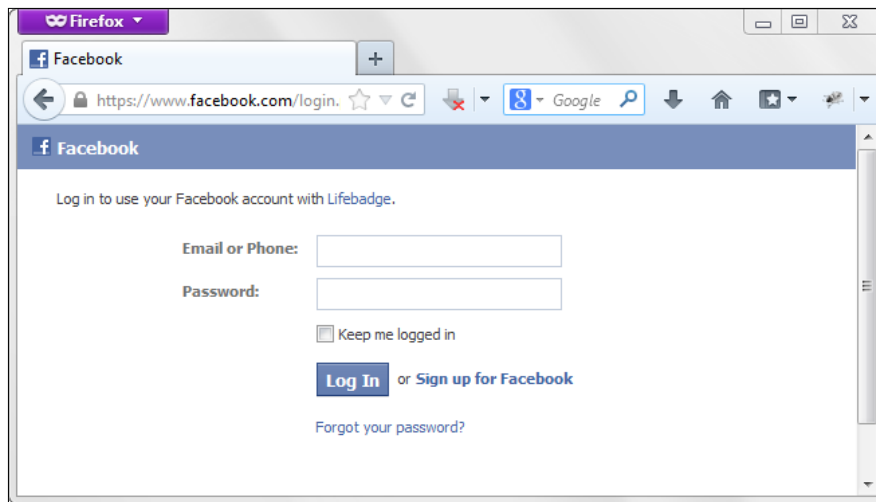
Authentication instances can have different levels of importance:

- i. **Required** means if valid credentials are not provided then authentication will fail.
 - ii. **Optional** means if valid credentials are not provided, then authentication is not marked as failed, but can be passed to another authentication module.
 - iii. **Requisite** means if valid credentials are provided then authentication will pass to the next authentication module. If valid credentials are not presented then authentication will fail without the option to try another authentication module.
 - iv. **Sufficient** means if valid credentials are provided, then authentication has been successful, and no other modules will be processed.
3. For our prototype we've switched both of the instances to **SUFFICIENT**, as shown in the preceding screenshot, which means that either of these authentication modules are enough by themselves to authenticate a user.
 4. There are also options to redirect users to either a **Successful Login URL** or a **Failed Login URL**. For our prototype we'll leave them blank.
 5. Click on the **Save** button and then the **Back to Authentication** button.

Testing our OAuth Client against Facebook as an OAuth Provider

Now that we've configured everything, we should test our configuration as follows:

1. First step is to head to a Policy Agent protected URL, for me this was `http://tomcat.kenning.local:9080/docs/`.



2. We should be redirected from there to Facebook as per our Authentication Chaining to use Facebook as the first authentication module. The URL I was redirected to was `https://www.facebook.com/login.php?api_key=187529064617026&skip_api_login=1&display=page&cancel_url=http%3A%2F%2Fopenam.kenning.co.nz%3A8080%2Fopenam%2Foauth2c%2FOAuthProxy.jsp%3Ferror_reason%3Duser_denied%26error%3Daccess_denied%26error_description%3DThe%2Buser%2Bdenied%2Bbyyour%2Brequest.&fbconnect=1&next=https%3A%2F%2Fwww.facebook.com%2Fdialog%2Fpermissions.request%3F_path%3Dpermissions.request%26app_id%3D187529064617026%26client_id%3D187529064617026%26redirect_uri%3Dhttp%253A%252F%252Fopenam.kenning.co.nz%253A8080%252Fopenam%252Foauth2c%252FOAuthProxy.jsp%26display%3Dpage%26response_type%3Dcode%26perms%3Demail%252Cread_stream%26fbconnect%3D1%26from_login%3D1&rcount=1`.
3. So what does all that mean? Let's put it through a URL decoder and format it nicely:
 - `https://www.facebook.com/login.php`
 - `api_key=187529064617026`
 - `skip_api_login=1`
 - `display=page`
 - `cancel_url=http://openam.kenning.co.nz:8080/openam/oauth2c/OAuthProxy.jsp?error_reason=user_denied&error=access_denied&error_description=The+user+denied+your+request.&fbconnect=1`

- `next=https://www.facebook.com/dialog/permissions.request?_path=permissions.request&app_id=187529064617026&client_id=187529064617026&redirect_uri=http%3A%2F%2Fopenam.kenning.co.nz%3A8080%2Fopenam%2Foauth2c%2FOAuthProxy.jsp`
- `display=page`
- `response_type=code`
- `perms=email%2Cread_stream`
- `fbconnect=1`
- `from_login=1`
- `rcount=1`

4. Some key values are listed as follows:

- `api_key` is the value provided to us by Facebook from our App and loaded into our OAuth authentication module.
- `cancel_url` will return us to our **Site URL** with some error codes as GET variables that we could parse and process.
- `next` forwards us to a Facebook dialog permission page which would request the Facebook App to have permission to access our information if we hadn't already agreed. This will then redirect us to our **Site URL**.
- `perms` lists the permissions we requested, **email**, and `read_stream`.

5. Finally after logging in we should be directed to our protected content.

Summary

In this chapter we learned about OAuth Providers and Clients, created a Facebook App, configured the Facebook App to be an OAuth Provider, configured OpenAM to be an OAuth Client, and used authentication chaining to allow us to access our Policy Agent protected content using a Facebook login. In the next chapter we will cover Two Factor Authentication.

8

Two Factor Authentication

In the last chapter we covered OAuth as a way of integrating our OpenAM installation to an existing cloud based identity provider. This is a great way of providing a good experience to users, especially on mobile devices. But sometimes there is a requirement for a higher standard of authorization requiring the use of two factor authentication. This chapter will describe setting up two factor authentication in OpenAM.

Understanding two factor authentication

Two factor authentication refers to the use of a secondary identity credential used for authentication. Typically this is a unique token generated by hardware or software.

Two factor authentication is used when there is a requirement for greater than normal authentication confidence. A typical example is Internet banking. Perhaps viewing your account balance requires just a normal username and password, but paying someone online requires two factor authentication.

Understanding OATH and how it relates to OpenAM

OATH is the method used to do two factor authentication in OpenAM. Because it is a common standard, there are many OATH token generators available, including clients for iPhone and Android.

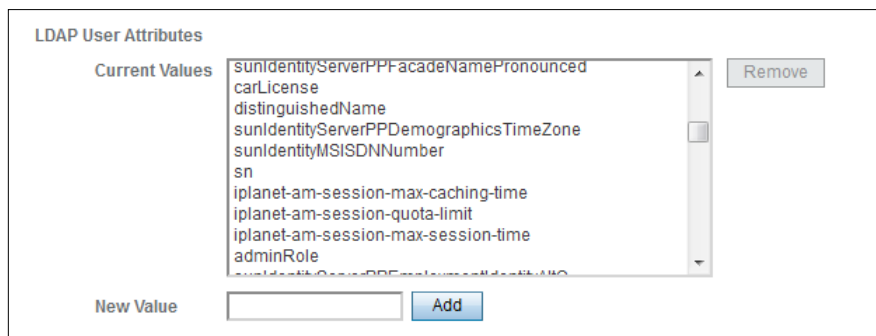
OATH can either use a static token that increments each time it is used (known as HOTP), or a token that changes after a predetermined amount of time (known as TOTP). A popular example of a TOTP token generator is the **Google Authenticator**.

Configuring OpenAM for two factor authentication

We'll need to configure OpenAM to access two new attributes from our LDAP directory, populate the directory with values for those attributes, and then configure the OATH authentication module, as well as changing our authentication chain.

Configuring OpenAM to use additional LDAP attributes

1. Log in to OpenAM Console. Click on the **Access Control** tab on the top and then click on the link to / (**Top Level Realm**).
2. Click on the **Data Stores** tab and then click on the embedded link to access the embedded OpenDJ LDAP server:



The screenshot shows the 'LDAP User Attributes' configuration page. On the left, under 'Current Values', there is a list of attributes: sunIdentityServerPPFacadeNamePronounced, carLicense, distinguishedName, sunIdentityServerPPDemographicsTimeZone, sunIdentityMSISDNNumber, sn, iplanet-am-session-max-caching-time, iplanet-am-session-quota-limit, iplanet-am-session-max-session-time, and adminRole. A 'Remove' button is located to the right of the list. At the bottom, there is a 'New Value' input field and an 'Add' button.

3. Scroll down to the **LDAP User Attributes** section. We need locations to store two values, a shared secret, and a shared counter value.

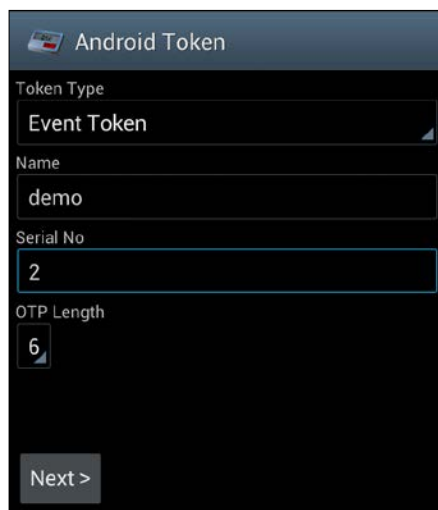
Because these values must be stored inside an LDAP directory, we need to conform to the schema that is applied to the LDAP directory. Therefore we must use attributes that already exist in the schema, unless we modify the schema ourselves.

4. For our prototype we'll use two existing attributes called `carLicense` for our shared secret and `roomNumber` for our shared counter.
5. Enter the attribute name into the **New Value** box and click on the **Add** button. Repeat for the other attribute.
6. Scroll to the top of the page and click on the **Save** button.

Installing an OATH HOTP token generator

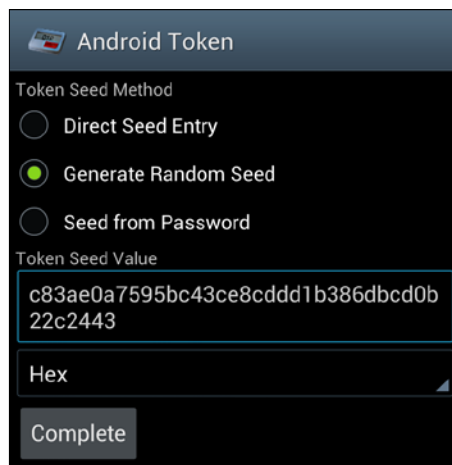
Install an OATH token generator by searching your phone's application store:

1. Since I use Android, I searched for OATH and found the **Android Token** application as shown in the following screenshot, and installed it:



The screenshot shows the 'Android Token' application interface. It has a dark theme with a blue header bar containing the app icon and title. Below the header, there are four input fields: 'Token Type' with a dropdown menu showing 'Event Token', 'Name' with the text 'demo', 'Serial No' with the text '2', and 'OTP Length' with a dropdown menu showing '6'. At the bottom left, there is a 'Next >' button.

2. Enter a token name, which should be the name of the user account. I used the value `demo` as shown in the preceding screenshot. For the value **Serial No or Counter**, I used the value 2 but feel free to use any other value. Set the **OTP Length** option to 6. Click on the **Next** button and you will be presented by the following screenshot:

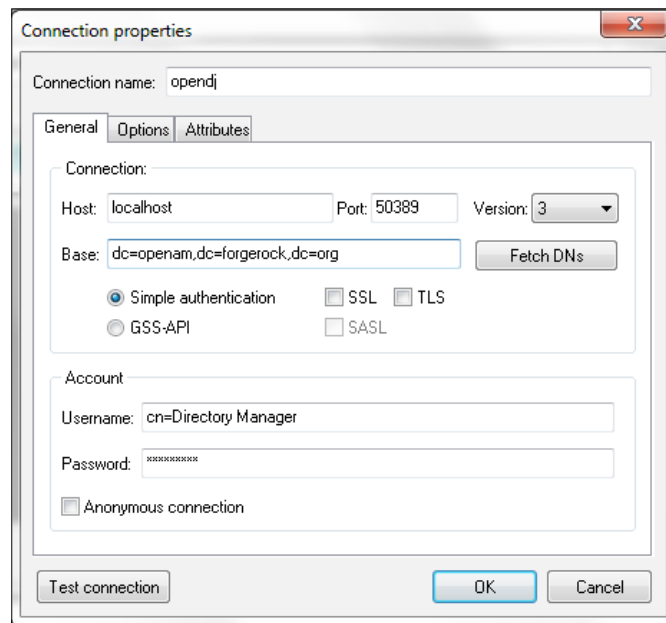


The screenshot shows the 'Android Token' application interface after clicking 'Next'. It has a dark theme with a blue header bar containing the app icon and title. Below the header, there are three radio button options for 'Token Seed Method': 'Direct Seed Entry', 'Generate Random Seed' (which is selected), and 'Seed from Password'. Below these options is a text field for 'Token Seed Value' containing the hexadecimal string 'c83ae0a7595bc43ce8cddd1b386dbcd0b22c2443'. Below the text field is a dropdown menu showing 'Hex'. At the bottom left, there is a 'Complete' button.

3. Next, we need to generate a shared secret. Copy that value, as well as the value of the counter in the previous step. Click on the **Complete** button.

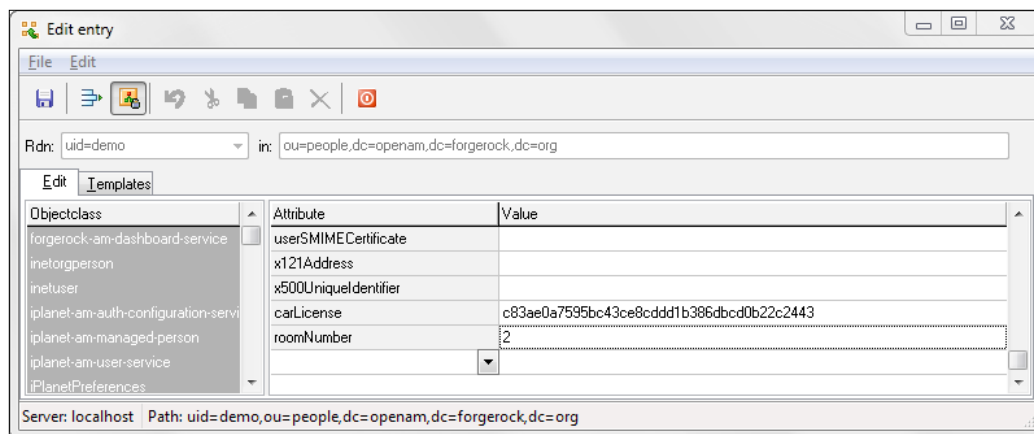
Populating our LDAP attributes with values

Now that OpenAM is aware of additional attributes in our LDAP directory, we need to populate those attributes.



1. Download an LDAP Browser. I used the open source LDAP Admin available for free download from <http://sourceforge.net/projects/ldapadmin/?source=dlp>.

- In your LDAP Browser, set the hostname to the name of your OpenAM instance. I used `localhost` as shown in the preceding screenshot. Set the port to `50389`, which is a specific port for the embedded instance of OpenDJ that comes with OpenAM. If you're using a standalone LDAP server you'll likely use port `389`. For the **username** use the value `cn=Directory Manager`. For the **password** use the password for the `amAdmin` account. My password as `adminpass`. You should be able to connect to the LDAP server as long as your OpenAM server is running.



- Locate the user you want to log in with. For me it was the **demo** user. Add two values, `carLicense` which will be storing our shared secret, and `roomNumber` which will be storing our shared counter as shown in the preceding screenshot.
- Populate those values with the values we gained from our token generator application.

Configuring the OATH authentication module

- In OpenAM from the main page, click on the **Access Control** tab, then click on the link to **(Top Level Realm)**. Click on the **Authentication** tab and scroll down to **Module Instances**.

2. Click on the link to the **OATH** module.

OATH Save Reset Back to Authentication

* Indicates required field

Realm Attributes

* Authentication Level:
The authentication level associated with this module.

* One Time Password Length:
The length of the generated OTP in digits. Must be 6 digits or longer.

Minimum Secret Key Length:
Number of hexadecimal characters allowed for the Secret Key.

* Secret Key Attribute Name:
The name of the attribute in the user profile to store the user secret key.

* OATH Algorithm to Use: ☒ HOTP ☐ TOTP
Choose the algorithm your device uses to generate the OTP.

HOTP Window Size:
The size of the window to resynchronize with the client.

Counter Attribute Name:
The name of the attribute in the user profile to store the user counter. This is required if HOTP is chosen as the OATH algorithm.

Add Checksum Digit: ☒ No ☐ Yes
This adds a checksum digit to the OTP.

Truncation Offset:
This adds an offset to the generation of the OTP.

TOTP Time Step Interval:
The TOTP time step in seconds that the OTP device uses to generate the OTP.

TOTP Time Steps:
The number of time steps to check before and after receiving a OTP.

Last Login Time Attribute:
Attribute to store the time of the users last login. This is required if TOTP is chosen as the OATH algorithm.

3. One of the values that needs changing is the **Secret Key Attribute Name** which is what we configured at the start of the chapter. Enter the value `carLicense` as shown in the preceding screenshot.
4. The other value that needs changing is **Counter Attribute Name**. Set this to `roomNumber`. Click on the **Save** button, then the **Back to Authentication** button.
5. Next, scroll to the **Authentication Chaining** section and click on the **IdapService** chain.

Do note that it is recommended to create a new authentication chain in production environments.

The screenshot shows a configuration window titled "(2 Item(s))" with buttons for "Add", "Remove", and "Reorder". Below these buttons is a table with three columns: "Instance", "Criteria", and "Options".

Instance	Criteria	Options
DataStore	REQUISITE	
OATH	REQUIRED	

6. Add the two authentication modules as shown in the preceding screenshot, with **DataStore** at the top as a requisite module, and **OATH** as a required module, specifically in this order. Click on the **Save** button, then the **Back to Authentication** button.

Testing two factor authentication

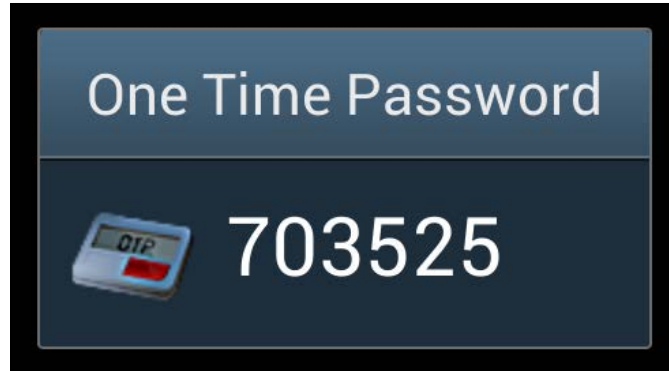
Now we will test our two factor authentication by logging in.

The screenshot shows the OpenAM login page. On the left is the OpenAM logo. On the right, the text "Sign in to OpenAM" is displayed above two input fields: "User Name:" with the value "demo" and "Password:" with masked characters. Below these fields is an orange "Log In" button.

1. Head to the OpenAM login page.
2. Enter your demo username and password. Click on the **Log In** button.

The screenshot shows the OpenAM OATH Authentication page. On the left is the OpenAM logo. On the right, the text "This server uses OATH Authentication" is displayed above a "One Time Password:" input field with masked characters. Below this field is an orange "Submit OTP Code" button.

The OATH page should appear as shown in the preceding screenshot.



3. Enter the one time password value as presented in your token application as shown in the preceding screenshot.

If everything has gone well, we should see information about our demo user as shown in the preceding screenshot.

Summary

In this chapter we learned about two factor authentication, specifically the OATH protocol used by OpenAM. We then configured OpenAM to use OATH to access one time passwords. We installed an OATH token generator on an Android phone, and then configured OpenAM to use the shared secret and counter from that application. In the next and final chapter we will cover Adaptive Risk authentication.

9

Adaptive Risk Authentication

In the last chapter we covered two factor authentication as a way of providing a higher level of authentication to our users. However, we shouldn't always burden users with additional authentication if it isn't needed. OpenAM can determine the risk of particular access requests and then provide higher or lower levels of authentication using **Adaptive Risk authentication**.

Understanding Adaptive Risk authentication

Adaptive Risk authentication allows OpenAM to determine the risk of a particular authentication, and decide whether additional authentication steps are required due to the risk.

A popular example is banking. If a user has logged in for the first time using a valid username and password then that's fine. But what if it's the first time from a particular IP address? That introduces a higher level of risk that can be mitigated by requesting two factor authentication. However, once that IP address has been validated with two factor authentication, it should become a trusted IP. This would mean we wouldn't request two factor authentication the next time the user logs in from that IP address.

Understanding how Adaptive Risk authentication works

The **Adaptive Risk** module has a risk threshold that is set manually, and by default is set to 1. There are a variety of different authentication risks which are each given a score. If the value of the score meets or exceeds the risk threshold, then the authentication fails.

If the Adaptive Risk module was set up to be a sufficient authentication module, and if the risk does not reach the risk threshold, then it will be a valid authentication and will allow the user, access to the resource. If the risk meets or exceeds the risk threshold, then the authentication fails and moves on to the next authentication method, typically two factor authentication.

Adding the Adaptive Risk module

1. Log in to the OpenAM console.
2. On the **Access Control** tab, click on the link / (Top Level Realm).
3. On the **Authentication** tab, scroll down to **Module Instances** and click on the **New** button and you will see a screen as follows:

The screenshot shows a web browser window with the OpenAM console interface. The browser address bar shows the URL `openam.kenning.co.nz:8080/openam/authentication/Au`. The page header includes a 'VERSION' dropdown, a 'LOG OUT' button, and user information: 'User: amAdmin Server: Waylon-PC'. The main content area is titled 'New Module Instance' and contains a form with the following fields:

- Name:** A text input field containing 'AdaptiveRisk'.
- Type:** A radio button selection list with the following options: Active Directory, Adaptive Risk (selected), Anonymous, Certificate, Data Store, Federation, HOTP, HTTP Basic, JDBC, LDAP, Membership, MSISDN, OATH, OAuth 2.0, RADIUS, SAE, SecuriD, Windows Desktop SSO, Windows NT, and WSSAuth.

At the top right of the form are 'OK' and 'Cancel' buttons. A red asterisk (*) indicates required fields.

Enter a name for your Adaptive Risk module. I called mine `AdaptiveRisk`. Select **Adaptive Risk** as the module type and click on the **OK** button.

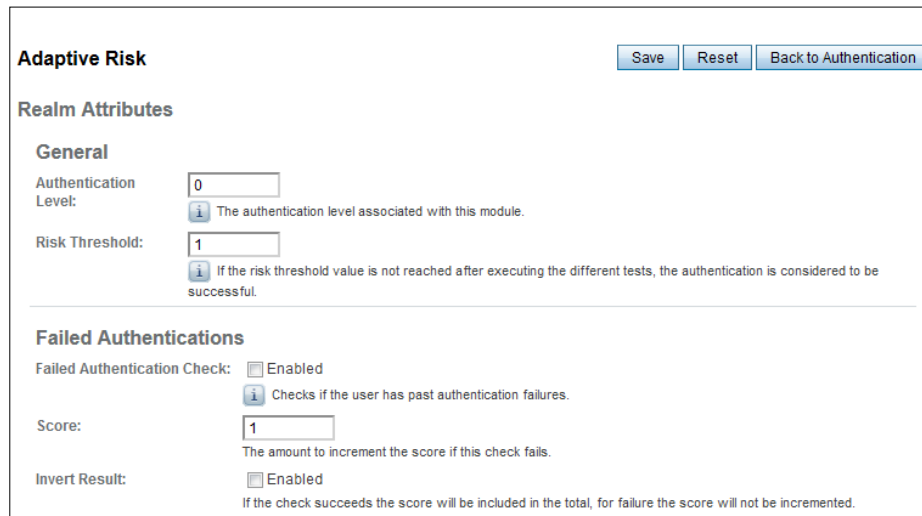
Module Instances (9 Items)

NewDelete

<input checked="" type="checkbox"/> <input type="checkbox"/>	Name	Type
<input checked="" type="checkbox"/>	AdaptiveRisk	Adaptive
<input type="checkbox"/>	DataStore	DataStore
<input type="checkbox"/>	Facebook	OAuth
<input type="checkbox"/>	Federation	Federation
<input type="checkbox"/>	HOTP	HOTP
<input type="checkbox"/>	LDAP	LDAP
<input type="checkbox"/>	OATH	OATH
<input type="checkbox"/>	SAE	SAE
<input type="checkbox"/>	WSSAuthModule	WSSAuthModule

Configuring the Adaptive Risk module

1. Click on the link to the Adaptive Risk module you created.



Adaptive Risk

Realm Attributes

General

Authentication Level:
The authentication level associated with this module.

Risk Threshold:
If the risk threshold value is not reached after executing the different tests, the authentication is considered to be successful.

Failed Authentications

Failed Authentication Check: ☒ Enabled
Checks if the user has past authentication failures.


Score:
The amount to increment the score if this check fails.

Invert Result: ☐ Enabled
If the check succeeds the score will be included in the total, for failure the score will not be incremented.

- **Risk Threshold** is the value required to be met or exceeded for authentication to fail. By default this is set to 1 which means only one of the checks ahead has to fail before the user needs to use another type of authentication.

- **Failed Authentications** examines if the account has failed authentication in the past. This will only work if **Account Lockouts** are enabled. **Invert Result** means apply the opposite for the rule, which would mean trigger this check if the user had not had a failed authentication.


IP Address Range

IP Range Check: ☐ Enabled
 Enables the checking of the client IP address against a list of IP addresses.

IP Range

Current Values


New Value

 The list of IP address to compare against the client IP address.


Score:
The amount to increment the score if this check fails.


Invert Result: ☐ Enabled
If the check succeeds the score will be included in the total, for failure the score will not be incremented.

IP Address History

IP History Check: ☐ Enabled
 Enables the checking of client IP address against a list of past IP addresses.

History size:
The number of client IP addresses to save in the history list.

Profile Attribute Name:
 The name of the attribute used to store the IP history list in the data store.





Save Successful IP Address: ☐ Enabled
 The IP History list will be updated in the data store

Score:
The amount to increment the score if this check fails.

Invert Result: ☐ Enabled
If the check succeeds the score will be included in the total, for failure the score will not be incremented.

- **IP Address Range** checks to see if the request came from a certain IP address range. If it does, then add the score of 1 to the check, which by default will trigger the threshold. The **Invert Result** option here is useful because we can say, unless results are coming from this known good IP range, then force users to do two factor authentication.


2. Configure these values to be your IP address range and select the **Invert Result** tick box. This will mean that if your IP address is in the above range then the risk score should increase by one, and therefore cause authentication to fail.
 - **IP Address History** looks at the IP Address history of previous logins. This would be useful in using the **Inverted Result** option, which would mean if we hadn't seen this address before, add a score of 1 to the check. Once the user has done the two factor authentication for that IP once, then no longer burden the user for two factor authentication.

Known Cookie	
Cookie Value Check:	<input type="checkbox"/> Enabled  Enables the checking of a known cookie value in the client request
Cookie Name:	<input type="text"/> <small>The name of the cookie to set on the client.</small>
Cookie Value:	<input type="text"/> <small>The value to be set on the cookie.</small>
Save Cookie Value on Successful Login:	<input type="checkbox"/> Enabled  The cookie will be created on the client after successful login
Score:	<input type="text" value="1"/> <small>The amount to increment the score if this check fails.</small>
Invert Result:	<input type="checkbox"/> Enabled <small>If the check succeeds the score will be included in the total, for failure the score will not be incremented.</small>
Device Cookie	
Device Registration Cookie Check:	<input type="checkbox"/> Enabled  Enables the checking of the client request for a known cookie.
Cookie Name:	<input type="text" value="Device"/> <small>The name of the cookie to be checked for (and optionally set) on the client request</small>
Save Device Registration on Successful Login:	<input type="checkbox"/> Enabled  Set the device cookie on the client response
Score:	<input type="text" value="1"/> <small>The amount to increment the score if this check fails.</small>
Invert Result:	<input type="checkbox"/> Enabled <small>If the check succeeds the score will be included in the total, for failure the score will not be incremented.</small>

- **Known Cookie** would be examining a cookie on the user's machine. This could be a value set by another web application on the same domain.


- **Device Cookie** is a check to see if a user is from a known and trusted device, which is determined by use of a per-device cookie.

Time Since Last Login

Time since Last login Check: ☐ Enabled
 Enables the checking of the last time the user successfully authenticated.

Cookie Name:
The name of the cookie used to store the time of the last successful authentication.


Max Time since Last login:
The maximum number of days that can elapse before this test.

Save time of Successful Login: ☐ Enabled
 The last login time will be saved in a client cookie

Score:
The amount to increment the score if this check fails.

Invert Result: ☐ Enabled
If the check succeeds the score will be included in the total, for failure the score will not be incremented.

Profile Attribute

Profile Risk Attribute check: ☐ Enabled
 Enables the checking of the user profile for a matching attribute and value.

Attribute Name:
The name of the attribute to retrieve from the user profile in the data store.


Attribute Value:
The required value of the named attribute.


Score:
The amount to increment the score if this check fails.


Invert Result: ☐ Enabled
If the check succeeds the score will be included in the total, for failure the score will not be incremented.

- **Time Since Last Login** would be a method of reducing the risk temporarily based on when the user last logged in. Do note that this check is done on a per day basis.
- **Profile Attribute** looks at the risks associated with a user's particular profile. An example of this being used would be for an administrator account. Because an administrator account can do more, it is of higher risk, so administrators should always be forced to use two factor authentication.

Geo Location

Geolocation Country Code Check: ☐ Enabled
 Enables the checking of the client IP address against the geolocation database.


Geolocation Database location:
 The path to the location of the GEO location database.

Valid Country Codes:
 The list of country codes that are considered as valid locations for client IPs.

Score:
The amount to increment the score if this check fails.

Invert Result: ☐ Enabled
If the check succeeds the score will be included in the total, for failure the score will not be incremented.

Request Header

Request Header Check: ☐ Enabled
 Enables the checking of the client request for a known header name and value.

Request Header Name:
The name of the required HTTP header

Request Header Value:
The required value of the named HTTP header.

Score:
The amount to increment the score if this check fails.

Invert Result: ☐ Enabled
If the check succeeds the score will be included in the total, for failure the score will not be incremented.

- **Geo Location** uses a geolocation database that is provided by Maxmind at <http://www.maxmind.com/app/country>. Do note that looking at locations based on IP is easily gotten around through the use of VPNs and other technologies so shouldn't be relied on in isolation. This is where chaining multiple checks together with different scores would provide you with a more sophisticated risk profile.
 - The final check available is **Request Header**. This looks to see if an HTTP header contains a known value.
3. Once these settings have been configured, click on the **Save** button.

Adding adaptive risk to the authentication chain

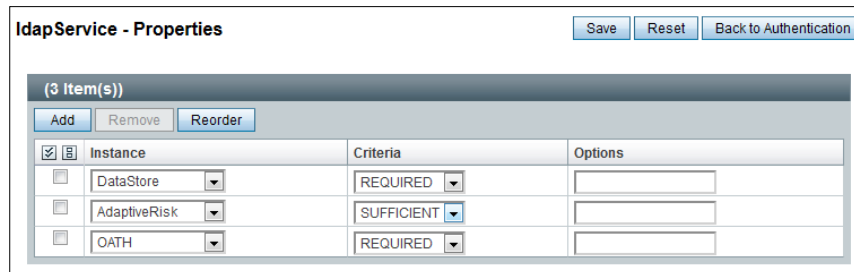
Now we will add our Adaptive Risk module to the default Authentication Chain.



On the **Authentication** tab, scroll down to **Authentication Chaining** and click on the link to your Authentication Chain. Mine is called `ldapService`.



Do note that it is recommended to create a new authentication chain in production environments.



In the preceding screenshot the first authentication method is the **DataStore** which will ask the user for a username and password. This is a required authentication method, which means the user must enter this correctly to move on to the next section.

The **Adaptive Risk** module is set as sufficient. This means that if the risk in the module is less than the risk score, then the authentication is successful and the user is granted access to the resources. If the risk is higher than the risk score, then authentication fails and the next authentication module is invoked, in this case **OATH** or two factor authentication.

With all this configured, a user accessing a protected resource should be asked to enter a username and password. If their IP address was in the list we specified, then authentication should fail, and the user is prompted as part of the OATH authentication module to enter their one time password.

Potential authentication patterns

The great thing about authentication chaining is that you can create very specific profiles to suit very specific authentication requests.

One example could be an Intranet. The Intranet could use Windows authentication in the first instance, then an Adaptive Risk module triggered to fail if requests are from outside a certain IP address range, which would then invoke LDAP authentication as well as two factor authentication.

Also note it is possible to have multiple Adaptive Risk modules in the same Authentication Chain. This would allow for the layered creation of authentication which becomes more burdensome as the risk increases.

However, it is important to note that authentication is a burden for users, and especially in the mobile world, we should strive towards the least possible authentication for the least possible risk, and only request additional authentication where necessary to reduce risk to an acceptable level.

Summary

In this chapter we learned about the Adaptive Risk module, which allows us to craft authentication that changes depending on the risk profile of the access request. We looked at the different types of filters available, and configured the IP address one as an example. Finally we looked at some potential patterns, and raised a caution around introducing too much authentication burden to your users.

We're now at the end of our book *Open Source Identity Management Principles and Patterns* using OpenAM 10.x. We learned to install OpenAM, configure it, created multiple instances of Tomcat, Apache, and authenticated against different data sources including Facebook.

OpenAM is professional enterprise quality software. The skills you've learned by touching and experiencing this software are using in enterprises as Identity Management becomes increasingly more important, especially with the trends of enterprise single sign on, and the growth of multiple devices.

Congratulations for learning, well done, and just remember that this is a taster on the Identity Management. I look forward to reading your blogs, books, and vlogs on the Identity Management one day.

Index

A

- Access Control tab** 80, 83, 90
- Adaptive Risk authentication**
 - about 89
 - working 89, 90
- Adaptive Risk module**
 - adding 90, 91
 - adding, to authentication chain 96
 - configuring 91-95
- Add button** 80
- Apache 2.4 local domain website**
 - Apache Policy Agent profile, creating in OpenAM 28
 - securing 28
 - securing, OpenAM Policy Agent used 30
- api_key value** 78
- authentication chain**
 - Adaptive Risk module, adding to 96
- Authentication Chaining**
 - configuring 75, 76
 - levels, optional 76
 - levels, required 76
 - levels, requisite 76
 - levels, sufficient 76
- Authentication Chaining section** 75
- authentication patterns** 97
- Authentication tab** 96

B

- Back to Authentication button** 85

C

- cancel_url value** 78
- CDSSO** 27
- CDSSO Domain List section** 36
- Circle of Trust** 58
- Complete button** 82
- Configuration Complete dialog** 24
- Configuration Directory** 21
- Configure button** 50, 66
- Cookie Domain** 20
- Create Hosted Identity Provider button** 49
- Cross Domain Single Sign On**
 - Tomcat Agent Profile, sharing 35
- Cross-Domain Single Sign On.** *See* CDSSO
- Cross Domain SSO option** 36

D

- Demilitarized Zone (DMZ)** 38
- device cookie** 94
- distributed authentication**
 - about 37
 - defense-in-depth architectures 38
 - OpenAM, preparing 38-40
 - policy agents communication, with OpenAM 37
 - testing 44-46
- distributed authentication application**
 - configuring 41-44
 - screenshot 42

distributed authentication
 application server

 configuring 41

downloading

 OpenAM 10.x 15

F

Facebook

 preparing, as OAuth Provider 70

failed authentication 92

Federation tab 63

Fedlet application server

 configuring 48

Fedlets

 about 47, 55

 creating 50-52

 comparing, with Policy Agents 47, 48

Fedlet setup

 validating 53, 55

Fedlet.zip

 deploying, onto Java application server 52

G

geo location 95

Google Authenticator 79

H

High Availability 24

HOTP 79

hot swappable values 33

I

identity data stores 13

identity levels, examples

 federated identities 10

 pseudonymous identities 9

 trusted identities 10

 trusted identities, with multiple contexts 10

Identity Management

 about 7

 claims, relating to 8

 components 12

 defining 7

 identity contexts 8

 identity levels, examples 9

 importance 9

 working 10, 11

Identity Management, components

 identity data stores 13

 identity managers 13

 identity policy agents 12

 identity providers 12

 identity service providers 12

identity managers 13

identity policy agents 12

identity providers 12, 57

installation

 OpenAM 10.1.0 19-24

Inverted Result option 93

Invert Result option 92

IP address history 93

IP address range 92

J

Java Key Store (JKS) 59

Java Runtime Environment (JRE) 17

Java Software Development Kit (JDK) 17

K

known cookie 93

L

LDAP Admin

 URL 82

LDAP attributes

 OpenAM, configuring to use 80

 populating, with values 82, 83

ldapService 96

Lightweight Directory Access Protocol
 (LDAP) 13

Log In button 85

M

Maxmind

 URL 95

N

New button 90

Next button 81

next value 78

OpenAM 10.1.0
installing 19-24

O

OATH 79

OATH authentication module

configuring 83, 84

OATH HOTP token generator

installing 81, 82

OAuth

about 69

clients 69

providers 69

OAuth authentication module

configuring 70-74

OAuth Client

testing, as OAuth Provider 76-78

OAuth Provider

Facebook, preparing as 70

OK button 91

One Time Password (OTP) 11

OpenAM

Apache Policy Agent profile,

creating 28, 29

configuring, as SAML Identity

Provider 58-60

configuring, for SimpleSAMLphp

SP trust 65, 66

configuring, for two factor

authentication 80

configuring, to use additional

LDAP attributes 80

LDAP attributes, populating

with values 82, 83

OATH authentication module,

configuring 83, 84

OATH HOTP token generator,

installing 81, 82

preparing, for distributed authentication 38

prerequisites 16

used, for policy agent communication 37

OpenAM, prerequisites

fully qualified domain name,

creating 16, 17

JRE, installing 17

Tomcat application server, downloading 18

Tomcat for OpenAM, configuring 18, 19

OpenAM 10.x
 downloading 15
OpenAM Enterprise Download
 Stack page 15
OpenAM Policy Agent
 used, for Apache securing 30, 31
OTP Length option 81
out of memory errors 19

P

perms value 78
Platform Locale 21
Portecle 59
prerequisites, OpenAM
 fully qualified domain name,
 creating 16, 17
 JRE, installing 17
 Tomcat application server, downloading 18
 Tomcat for OpenAM, configuring 18, 19
profile attribute 94

R

risk threshold 91

S

SAML
 about 57
 Circle of Trust 58
 Identity Providers 57
 Identity Providers (IdP) 57
 Service Providers 58
 Service Providers (SP) 57
SAML Circle of Trust
 testing 66
SAML hosted identity provider
 creating 49, 50
SAML Identity Provider
 OpenAM, configuring as 58-60
Save button 80, 85, 95
Security Assertion Markup Language. *See*
 SAML
Server URL 20
Service Provider
 about 12
 SimpleSAMLphp, configuring as 63, 64

Service Providers 58
SimpleSAMLphp
 configuring, as Service Provider 62, 64
 installing 61, 62
Single Sign On (SSO) 9

T

Time Since Last Login 94
Tomcat
 configuring 31-33
 securing, with OpenAM Policy
 Agent 33, 34
Tomcat 6 remote domain website
 securing 31
 Tomcat, configuring 31
 Tomcat Policy Agent profile, creating 31
Tomcat Agent Profile
 configuring, for Cross Domain
 Single Sign O 35
Tomcat Policy Agent profile
 creating 31, 32
Top Level Realm link 31
TOTP 79
two factor authentication
 about 79
 OpenAM, configuring for 80
 testing 85, 86

W

WAR
 differentiating, with ZIP file 16



Thank you for buying **Open Source Identity Management Patterns and Practices Using OpenAM 10.x**

About Packt Publishing

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: www.packtpub.com.

About Packt Open Source

In 2010, Packt launched two new brands, Packt Open Source and Packt Enterprise, in order to continue its focus on specialization. This book is part of the Packt Open Source brand, home to books published on software built around Open Source licences, and offering information to anybody from advanced developers to budding web designers. The Open Source brand also runs Packt's Open Source Royalty Scheme, by which Packt gives a royalty to each Open Source project about whose software a book is sold.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to author@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.



OpenAM

ISBN: 978-1-849510-22-6

Paperback: 292 pages

Written and tested with OpenAM Snapshot 9 – the Single Sign-On (SSO) tool for securing your web applications in a fast and easy way

1. The first and the only book that focuses on implementing Single Sign-On using OpenAM
2. Learn how to use OpenAM quickly and efficiently to protect your web applications with the help of this easy-to-grasp guide
3. Written by Indira Thangasamy, core team member of the OpenSSO project from which OpenAM is derived



Oracle Identity and Access Manager 11g for Administrators

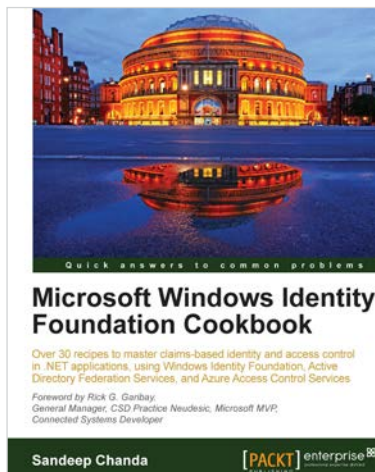
ISBN: 978-1-849682-68-8

Paperback: 336 pages

Administer Oracle Identity and Access Management, installation, configuration and day-to-day tasks

1. Full of illustrations, diagrams, and tips with clear step-by-step instructions and real time examples
2. Understand how to Integrate OIM/OAM with E-Business Suite, Webcenter, Oracle Internet Directory and Active Directory
3. Learn various techniques for implementing and managing OIM/OAM with illustrative screenshots

Please check www.PacktPub.com for information on our titles



Microsoft Windows Identity Foundation Cookbook

ISBN: 978-1-849686-20-4

Paperback: 294 pages

Over 30 recipes to master claims-based identity and access control in .NET applications, using Windows Identity Foundation, Active Directory Federation Services, and Azure Access Control Services.

1. Gain a firm understanding of Microsoft's Identity and Access Control paradigm with real world scenarios and hands-on solutions.
2. Apply your existing .NET skills to build claims-enabled applications.
3. Includes step-by-step recipes on easy-to-implement examples and practical advice on real world scenarios



Oracle Fusion Middleware Patterns

ISBN: 978-1-847198-32-7

Paperback: 224 pages

10 unique architecture patterns enabled by Oracle Fusion Middleware

1. First-hand technical solutions utilizing the complete and integrated Oracle Fusion Middleware Suite in hardcopy and ebook formats
2. From-the-trenches experience of leading IT Professionals
3. Learn about application integration and how to combine the integrated tools of the Oracle Fusion Middleware Suite - and do away with thousands of lines of code

Please check **www.PacktPub.com** for information on our titles