Strong Authentication and Security for Oracle Application Express

Giuseppe “Gippa” Paternò

Preface by Mark Shuttleworth

www.gpaterno.com
March 2011

Page intentionally
Left blank

Table of content

About this publication

The author

Giuseppe Paternò is known in the computer industry with the nickname of *Gippa*. He actively participates in the Italian security community, with particular regards to the famous “sikurezza.org”; as an expert in computer architecture, he works within the most famous Italian and foreign companies in the telecommunications, government and finance industries.

Giuseppe started participating in the development of the Italian telecommunication network since he was 15 years old by entering the world of the BBS. While in the BBS world, he started his roots in the Internet first through the BBS Galactica, and then being part of a small ISP named Datanord Multimedia. By that time (1995), he joined the “newly born” Italian Naming Authority for the operation and rules of the “.it” domains, along with the famous italian Internet founders, i.e. Allocchio Claudio, Daniele Vannozzi, Joy Marino and Marco Negri. Due to his works and the fewer and fewer technical content of the Naming Authority, he has decided to retire from active participation, although he’s still with the MAIL-ITA (ITA ex-PE).

It is in this time frame that Giuseppe entered the security world: some crackers compromised the servers under his control. He began to study the core behavior of the Cisco routers and Unix systems, most notably Linux, to protect his systems

In 1996 he joined IBM, where he worked as a specialist on AS/400 networking and Internet systems, followed by one year of assignment at Lotus Software in Dublin as a cross-platform specialist. It’s in Ireland that Giuseppe becomes aware of the mailing list “sikurezza.org” and knew some of the founders. The “Dublin period” is the one in which Giuseppe understood his attitude towards in-deep security.

Back to Italy in 2001, after a short period in Infostrada/Wind as a core Internet router/system senior administrator, Giuseppe was hired by Sun Microsystems as a Senior Network and Security Architect. He began to attend domestic and international big enterprise environments, such as telecommunication companies and financial institutes, working on important projects. From 2006 to 2010 became Solution Architect at Red Hat, with the task of being the focal point of security in EMEA.

He worked on important projects such as the creation of the standard for J2ME Over-The-Air (OTA) provisioning along with Vodafone, the architecture and standards for the delivery of MHP applications through the digital terrestrial television (DTT) on behalf of DTT Lab (Telecom Italia/LA7) and implementation of HLR for Vodafone landline services.

His project of protecting confidential files in a government agency will lead him to SMAU in 2008 with the speech “Protecting confidential files from unauthorized use with SE-Linux, a case study”: whitepapers and slides which are available on this website.

In June 2009 he was appointed Visiting Researcher at the University of Dublin

Trinity College at the Center for Telecommunications Value-Chain Research (CTVR).

He's currently contracted as EMEA OEM Technical Manager by Canonical in the OEM Engineering team.

More information is available on site and *http://www.gpaterno.com/* on LinkedIn *http://www.linkedin.com/in/gpaterno*.

Copyright

Disclaimer

I do not speak on behalf of my employer and/or contracting parties, nor am I authorized to represent them publicly. All and any opinion and results expressed in this document are solely mine and do not represent my employer and/or contractor point-of-view. All the tests and any project contribution are done as an independent researcher out of business hours.

Preface

As business moves into the all-digital era, security and assurance have become even more important. Attacks on core infrastructure are no longer mere graffiti - they are commercial crimes, with sophisticated actors and profound consequences. A deep understanding of strong authentication and security is valuable for any administrator, but for those who find themselves custodians of critical assets or data, it is essential.

Mark Shuttleworth

Founder and Product Strategist of
Canonical Ltd and Ubuntu Project.
Founder of Thawte.

Introduction

Laziness is probably the reason I never became a real developer. It's not the fact I cannot program, but the fact that I wish to access some data quickly and I do want to code as less as I can in order to achieve the result. I found Oracle Application Express very useful as a solution to address my needs, but it is missing a real Web Single Sign-On experience and I can't provide strong authentication via One Time Passwords (OTP) keys. Moreover, any application - that is hosted in the same installation- is exposed to anybody who can access the machine via a standard web browser: it means that anyone can potentially access the administrative and developer interface.

This publication aims at documenting the process to address the need for a stronger authentication methodology and for protecting the application from misuse. The target of this publication is an intermediate Linux system administrator, with some basic Oracle knowledge and web technologies (Apache web server and tomcat).

Oracle Application Express

Oracle Application Express is a web-based Rapid Application Development (RAD) environment that is built on top of an Oracle Database. Oracle Application Express is also known as Apex or HTMLDB. This product can scale from a few users, by leveraging Oracle Database 10g Express Edition, to hundreds of users with a full Oracle Database in a Real Application Cluster (RAC) configuration. As such, Apex is supported on a wide range of operating systems, most notably Unix/Linux and Windows.

I personally like Apex as it's a quick and easy way of creating a web interface to handle data: this is sometimes referred as Create/Read/Update/Delete (CRUD) application. Furthermore, it supports Linux and can be installed over Oracle Database Express Edition (XE), a freely available edition of the popular Oracle database, which is a good choice for a typical small and medium business scenario with a small budget. Apex over Oracle XE is easy to extend through the PL/SQL language and easy to back-up: it's as simple as creating an “export” of the tablespace. Although Apex has been used as a customer portal for a big Italian mobile telecommunication company, I do think it has some limitations due to the lack of extensibility, especially when used in a big enterprise application or in specific use cases out of CRUD applications. Those cases, however, can be easily addressed through languages such as Java, PHP, Ruby or Python.

Issues about Apex security

I perceived two limitations about Apex on the security area. This first problem is authentication; the other one is related to application protection.

Apex has some built-in authentication methods: no authentication, application-based credentials, database-based credentials, LDAP and Oracle Web Single Sign-On systems. My objective is to use stronger authentication methods to access private data, such as One Time Passwords (OTP) or X.509 digital certificates with smartcards. The built-in LDAP method can be a way to perform authentication, however it has some limitations[1] making it difficult to integrate if a custom LDAP schema is implemented, or if there's a need to access Microsoft Active Directory. Additionally, using LDAP directly won't be as flexible as having a common web-based single sign-on system. I guess that the Oracle official solution is buying their Web Single Sign-On product, however I doubt that small and medium businesses can afford to buy the product.

On the other hand, exposing an Apex application to the Internet, to an extranet or to an unprotected intranet can lead to some security issues. First and foremost is the fact that every Apex application is being exposed, including the administration and development interfaces, which are Apex application themselves. A typical case might be, for example, that a customer wishes to expose a single application to an Extranet and the remaining applications to the internal team. Apex is also sometimes prone to SQL injection and different security issues, as demonstrated by well-known security professionals[2].

1 Apex uses username replacement in an LDAP bind call, ex: "UID=%u, OU=myorg, DC=site, DC=com". This will not work if you have different organizational units, if you have a different DN in an existing LDAP, or with Windows Active Directory where you have a full name plus surname in the DN, instead of samaccountname.

2 There's an good number of blogs about each Apex vulnerability, however a good source about possible exploiting scenario is “Hacking and Hardening Oracle Express Edition” by Alexander Kornbrust.

The "Defense in Depth" philosophy

Defense has always been a technique, or better an art, since ancient Rome. In the medieval era, precious manuscripts were protected by several layers of defense: they were usually preserved in castles placed in the highest hills, surrounded by a ditch full of crocodiles, in the highest tower, in a secret room, protected by several traps and with poisons inside. Although an attacker may find it easier to breach the first layers, as he advances he continues to meet effective resistance. As he penetrates deeper, the attacker becomes vulnerable and, should the advance stall, the attacker himself risks being entrapped.

Although media, attackers and things to protect have changed, defense techniques still remain valid. In the IT field, secure systems and data must still be protected by using several security layers. We will call this philosophy *Defense in Depth*.

In brief, the Defense in Depth philosophy represents the use of multiple computer security techniques to mitigate the risk of a particular defense being compromised or circumvented. An example could be anti-virus software installed on individual workstations while there is a virus protection on all firewalls and servers within the same environment. Different security products may be on different systems within the network, helping to prevent a shortfall in any one defense leading to a wider failure. This philosophy should be applied on every system, especially on those that hold sensible data or whenever strict-security is required (military, government, finance, ...). Every system, configuration, application and even software development itself has to be thought in a secure way, i.e. giving the least privilege to the user/application, restricting who can do what.

Applying this philosophy to a computing environment means breaking security into the following components:

- Installing a minimized system
- Hardening the system
- Activating/configuring a mandatory access control system (if available) such as SE-Linux
- Use virtualization to confine the application
- Configuring the application securely
- Authenticating users
- Updating systems with the latest security patches

Throughout this document, we will follow the "defense in depth" philosophy.

The architecture

In order to implement the authentication method, I decided to leverage Jasig's Central Authentication Server (CAS) that allows different authentication methodologies. A Web Application Firewall (WAF), in particular ModSecurity[3], will be used to restrict access to a given apex application and protect from common web attacks. Figure 1 in page 12 shows the implemented architecture. The flow of information is:

1. The user connects to the given APEX application URL;
2. ModSecurity checks if the user is trying to access the allowed application and if it's an attack from a malicious user;
3. Apache CAS module is called to authenticate the user;
4. The user is redirected to CAS' authentication portal. If the user is using a digital certificate (Smartcard), then it is transparently validated and authenticated, otherwise he/she will be prompted for username and an OTP password from his/her token device;
5. (OTP only) CAS' radius module queries the OTP server through RADIUS to check the provided password;
6. The CAS module populates the "REMOTE_USER" environment variable;
7. The Apache PL/SQL gateway (mod_owa) connects to the Oracle database and executes the requested function or stored procedure;
8. A custom authentication schema decodes the "REMOTE_USER" environment variable and transforms it into a compatible Apex user.

3 ModSecurity is a popular Open Source Web Application Firewall, more on the website *http://www.modsecurity.org/*

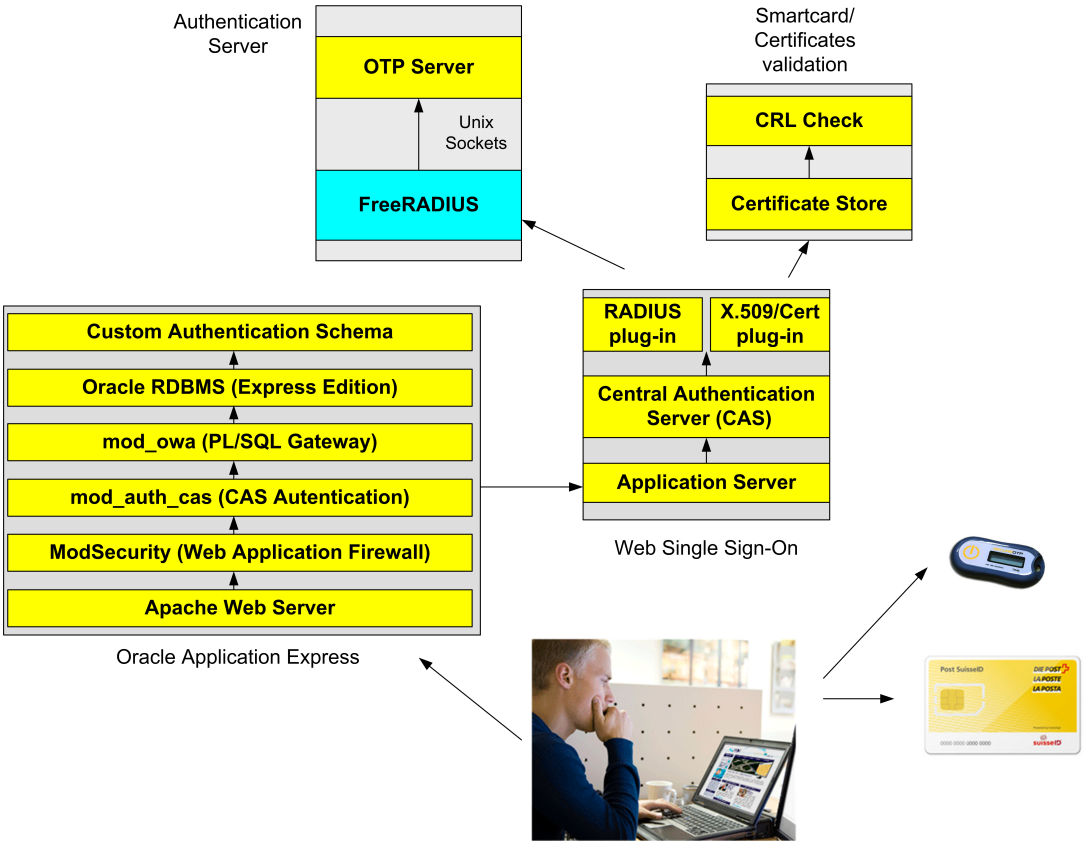

Figure 1 Architecture

Initial setup

The test environment was composed by three Linux servers, one will host the CAS authentication server, the second the Oracle Database Express Edition plus Application Express and the remaining with the One Time Password server.

The selected distribution was Ubuntu Linux 10.04 LTS[4] that was installed over a virtual infrastructure. Oracle made Express Edition available for Ubuntu: the advantage over other commercial Linux distributions is that the user is free to subscribe for support, while other distributions need a compulsory subscription agreement to have the software installed. Moreover, Ubuntu Linux installs itself by default with a minimized and hardened system, in full compliance with the "Defense in Depth" philosophy.

A lot of commercial One Time Password products are available, but for the OTP Server I selected the OTPD open source server I maintain[5]. The user is free to use any of his choice; the requirement is that the OTP server should be compliant with the RADIUS protocol. The solution has been tested also with smartcard authentication through the SuisseID, Switzerland's national electronic identity card.

This document will not cover basic Ubuntu Linux installation, as well as the OTP Server installation, Apex application development[6] and an LDAP system such as Microsoft Active Directory or OpenLDAP (optional for smartcard support).

4 LTS stands for "Long Term Support", i.e. the supported version of Ubuntu for the enterprises.

5 OTPD is available at *http://otpd.googlecode.com/*

6 For APEX development I suggest "Beginning Oracle Application Express" by Rick Greenwald and "Pro Oracle Application Express" by John E. Scott and Scott Spendolini. Oracle website as a lot of information as well, check it at http://apex.oracle.com/

The Central Authentication Service (CAS)

What is CAS

The Central Authentication Service (CAS) is a web-based single sign-on protocol that is somehow inspired by Kerberos.

CAS aims to provide a real single sign-on experience to an user, so that he/she can access multiple web applications while providing his/her credentials (such as userid and password) only once. It has several advantages, one is allowing web applications to authenticate users without gaining access to a user's security credentials, another other big advantage is abstracting security credentials from a web-based application: a security manager can decide to start requiring users to provide Active Directory credentials, while in a second stage he/she can decide to implement One Time Password or even SmartCard authentication without changes in the application.

CAS is not only as a protocol, but also an implementation: originally developed by the Yale University, is now an open source project of the Java Architectures Special Interest Group (JaSIG)[7].

The CAS protocol involves at least three parties: a client web browser, the web application requesting authentication, and the CAS server. When the client visits an application, which requires authentication, the application redirects the client to CAS. CAS validates the client's authenticity, usually by checking a username and password against a database (such as Kerberos, LDAP, Active Directory, etc...).

If the authentication succeeds, CAS returns the client to the application, passing along a security ticket. The application then validates the ticket by contacting CAS over a secure connection and providing its own service identifier and the ticket. CAS then gives the application trusted information about whether a particular user did authenticate successfully.

Configure CAS

The CAS framework is essentially a Java web application that is installed in the form of web archive (WAR) file. The CAS server, as downloaded from the Jasig website, is an archive that has the sources and a pre-made WAR file, but has only basic functionalities included. To have the required functionalities and maybe customized web pages, an ad-hoc build has to be made.

We'll have a quick review of the whole process. First of all, a JDK is needed, therefore partner repositories must be enabled in */etc/apt/sources.list* to install Sun Java JDK:

7 JaSIG CAS Home Page is *http://www.jasig.org/cas*

```
deb http://archive.canonical.com/ubuntu lucid partner
deb-src http://archive.canonical.com/ubuntu lucid partner
```

The next step is update the cache and install the Sun Java JDK as follows:

```
# apt-get update
# apt-get install -y sun-java6-jdk
```

The CAS sources are organized through the Apache Maven building tool. In order to rebuild the WAR file, maven must be installed on the system:

```
# apt-get install -y maven2
```

A Java web container to run the CAS server is also needed. In this laboratory, Apache Tomcat was chosen. The Ubuntu distribution has a package for tomcat, but I usually prefer to download the archive from the official website[8] and uncompress on the local filesystem. The Tomcat archive was uncompressed under */usr/local/* and a symlink was created for convenience:

```
# cd /usr/local
# ln -s apache-tomcat-6.0.30 tomcat
```

The next step is to download the CAS[9] source code and uncompress to a convenient directory:

```
# tar xzvf cas-server-3.4.5-release.tar.gz
```

and then modify the project file to add the needed features. To support OTP tokens we need the RADIUS plugin, while for the smartcard/digital certificate support we need the x509 module. The project file is *cas-server-webapp/pom.xml* and we need to add the following dependencies statement, before the *</dependencies>* tag:

```
<dependency>
     <groupId>${project.groupId}</groupId>
     <artifactId>cas-server-support-radius</artifactId>
     <version>${project.version}</version>
</dependency>
```

8 Apache Tomcat can be downloaded from *http://tomcat.apache.org/*
9 Jasig CAS can be downloaded from *http://www.jasig.org/cas/download*

```
<dependency>
     <groupId>${project.groupId}</groupId>
     <artifactId>cas-server-support-x509</artifactId>
     <version>${project.version}</version>
</dependency>
```

For a typical project, you might want to customize the JSPs that comes with the CAS server. This is the right phase to make such modifications. Once the modification are completed, we need to create the web archive and deploy it with the following commands:

```
# mvn -Dmaven.test.skip=true package install
# cp /root/.m2/repository/org/jasig/cas/cas-server-webapp/3.4.5/cas-server-webapp-3.4.5.war /usr/local/tomcat/webapps/cas.war
```

It is a good idea to test if the environment is working so far. As such, we do need to launch the Tomcat server through the */usr/local/tomcat/bin/startup.sh* script and connect to the following address:

http://cas-server.mydomain.com:8080/cas

Please note that the above *cas-server.mydomain.com* is the IP address of your CAS server. If the application was deployed correctly, you will be able to see the CAS login form, similar to the one in the Figure 2 (page 17). Always check for the *catalina.out* log file for any issue. In this phase, we can test authentication by using the same word as username as password, for example try "p" as username and "p" as password. This is due to the default authentication handler *SimpleTestUsernamePasswordAuthenticationHandler* that check that anything used as username will be used as password.

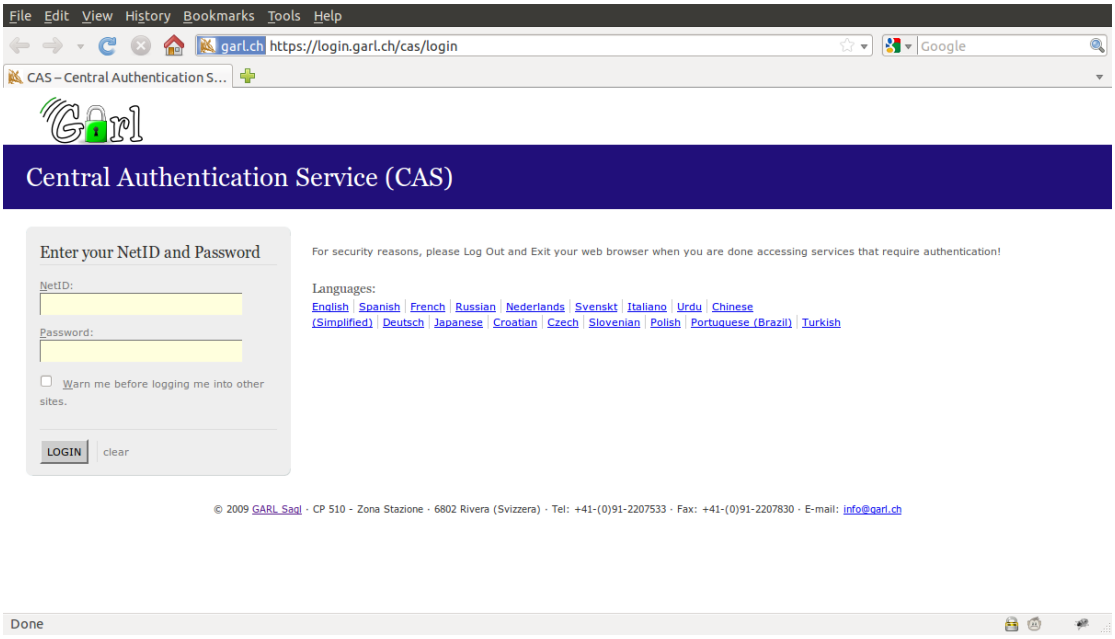

Figure 2 CAS Login

We will need to remove this handler, as it's very dangerous in production environments, and add the appropriate handlers for our authentication schema. To configure such handlers, we need to modify the *deployerConfigContext.xml* configuration file that can be found under the *cas/WEB-INF* directory. To add the RADIUS authentication handler, include the following lines within the *authenticationHandlers* property:

```
<bean
class="org.jasig.cas.adaptors.radius.authentication.handler.support.R
adiusAuthenticationHandler">
      <property
            name="servers">
            <list>
                  <bean

      class="org.jasig.cas.adaptors.radius.JRadiusServerImpl">
                        <constructor-arg index="0"
value="radius1.example.org" />
                        <constructor-arg index="1"
value="SHARED_SECRET" />
                        <constructor-arg index="2">
                              <bean

      class="net.jradius.client.auth.PAPAuthenticator" />
                        </constructor-arg>
                        <constructor-arg index="3" value="1812" />
                        <constructor-arg index="4" value="1813" />
```

```
                        <constructor-arg index="5" value="30" />
                        <constructor-arg index="6" value="3" />
                  </bean>


            </list>
      </property>
      <property
            name="failoverOnException"
            value="true" />
</bean>
```

Ensure that a proper configuration has been set in you OTP RADIUS server. In the lines above, replace *radius1.example.org* with your RADIUS IP address or host name and replace *SHARED_SECRET* with the actual shared secret as configured in the RADIUS server.

To test the above configuration we need to restart Apache Tomcat and connect to the following address:

http://cas-server.mydomain.com:8080/cas

The login prompt will appear; try to log-in using a defined username and the appropriate OTP password. If everything was set up correctly, you should be able to log-in the system. It is also important to modify the file */usr/local/tomcat/webapps/cas/WEB-INF/cas.properties* to reflect the appropriate CAS information (hostname, port, etc...) , as these information are exchanged with an application that is requiring access through CAS.

Also, ensure that CAS is running over SSL, either configuring Tomcat SSL connector[10] or by using Apache HTTPD and the *mod_jk* module[11].

10 A good reference to use Tomcat SSL connector could be http://tomcat.apache.org/tomcat-5.5-doc/ssl-howto.html

11 http://tomcat.apache.org/tomcat-3.3-doc/tomcat-ssl-howto.html#s4

Oracle Database and Apex

Install Oracle XE on Ubuntu Linux

Once the authentication part is completed, it is time to set up the Oracle part. The first step is to install Oracle on the designated Linux machine. Oracle made available Oracle Database Express (XE) 10g for debian-based distributions, thus also for Ubuntu. The installation is quite easy, first modify the */etc/apt/sources.list* file and append following line:

```
deb http://oss.oracle.com/debian unstable main non-free
```

This line will enable Oracle public repository into your installation. Before the actual installation, the step is to import Oracle's GPG key into the *apt* cache through the command:

```
# wget http://oss.oracle.com/el4/RPM-GPG-KEY-oracle -O- | \
apt-key add -
```

Then update the packages cache and install Oracle Database:

```
# apt-get update
# apt-get install oracle-xe
```

Once installed, a basic configuration is needed. You must configure Oracle TCP/IP listener port, HTTP port for Apex, the SYS and SYSTEM password, and if you want the Oracle service to be started at boot time. In order to do that, simply type the following command:

```
# /etc/init.d/oracle-xe configure
```

For this example, we will leave the default ports for the listener, i.e. TCP 1521, and the proposed Oracle Apex HTTP port, i.e. TCP 8080.

You must to set the Oracle environment variables in you in your profile file (eg: the file *~/.bash_profile*) or set a global profile by creating a file */etc/profile.d/oracle.sh* with the following content:

```
ORACLE_HOME=/usr/lib/oracle/xe/app/oracle/product/10.2.0/server
export ORACLE_HOME
ORACLE_SID=XE
export ORACLE_SID
NLS_LANG=`$ORACLE_HOME/bin/nls_lang.sh`
export NLS_LANG
```

```
PATH=$ORACLE_HOME/bin:$PATH
export PATH
```

The above variables will set the Oracle Home directory, i.e. where the server binaries are, the Oracle SID name (or instance name), the language. They also place the Oracle binary directory in the default search path.

By default, Oracle Apex will listen to the loopback interface only, i.e. 127.0.0.1, as it's supposed to be installed on a developer machine instead of a server. The service must be enabled to listen all available IP addresses; we will enable the service through the Oracle *sqlplus* utility:

```
$ sqlplus system@XE
SQL> EXEC DBMS_XDB.SETLISTENERLOCALACCESS(FALSE);
```

This setting will be immediately applied, so you can test if the Apex is reachable from an external system. From a browser, you can type the following URL:

http://myapex.domain.com:8080/apex/

Remember to use the "/apex/" suffix, because otherwise you will be redirected to a static page containing the Oracle license agreement. If you typed the right URL, you will be redirected to a general Apex administrative login page.

Please remember to set the listener local access back to TRUE when going in production or restrict access to the port through a firewall, as we will use Apache PL/SQL gateway to access the application.

Update Application Express

The Apex shipped with Oracle Database Express is an older release and has some security issues. We should upgrade Apex to the latest release in order to get the latest security updates, software enhancements and for the best possible user experience. As for the time of writing, Apex is in the 4.x version. An updated can be downloaded from the following URL:

http://www.oracle.com/technetwork/developer-tools/apex/downloads/index.html

The instructions available from the Oracle website[12] are quite clear, however let's summarize the necessary steps. The Apex update comes in the form of a zip file, for example: apex_4.0.2.zip. Upload the compressed archive to the server, go to a convenient directory, such as */usr/local*, and unzip the archive's contents.

12 Follow the instructions of Embedded PL/SQL on: *http://download.oracle.com/docs/cd/E17556_01/doc/install.40/e15513/otn_install.htm#insertedID4*

There are two kinds of upgrade procedures depending on the environment you are running, i.e. development and runtime. The development environment has the full Apex feature set available for developers, while the runtime has only a specific subset of Apex. In a production environment, we expect that the user will run the runtime environment, but in this paper we will describe the update of the development environment, as most of the people will test it into a development/pre-production environment. Please refer to the on-line Oracle documentation for the appropriate upgrade procedure for the runtime environment.

Change your current working directory to the location where Apex was unpacked, ex: */usr/local/apex/*, and run sqlplus to start the upgrade procedure:

```
$ sqlplus /nolog
SQL> CONNECT SYS as SYSDBA
```

Next step is to actually execute the upgrade procedure, please check that you are running the command inside the Apex directory:

```
SQL> @apexins SYSAUX SYSAUX TEMP /i/
```

Note that this procedure will take a very long time. In the command above, the "*/i/*" is the virtual path for the images: once the above procedure has finished, a separate procedure has to be executed to point the virtual image directory (the above *"/i/"*) to the real path on the filesystem, which in our example is set to the */usr/local/apex/images* directory:

```
$ sqlplus /nolog
SQL> CONNECT SYS as SYSDBA


SQL> @apxldimg.sql /usr/local
```

Note in the above command that the specified directory is the base apex directory, i.e. *"/usr/local"*: the procedure will automatically append the *"apex/images"* suffix to the specified directory.

The final step is to change the default administrative password with the following procedure:

```
SQL> @apxchpwd
```

To make sure that the change has been made, try to log in to the web interface with "INTERNAL" as workspace, "ADMIN" as user and the just-defined password, as for example in the Figure 3 in page Figure 3.

Now it's time to set-up a workspace, and create your own application or use the example provided with the newly-created workspace.

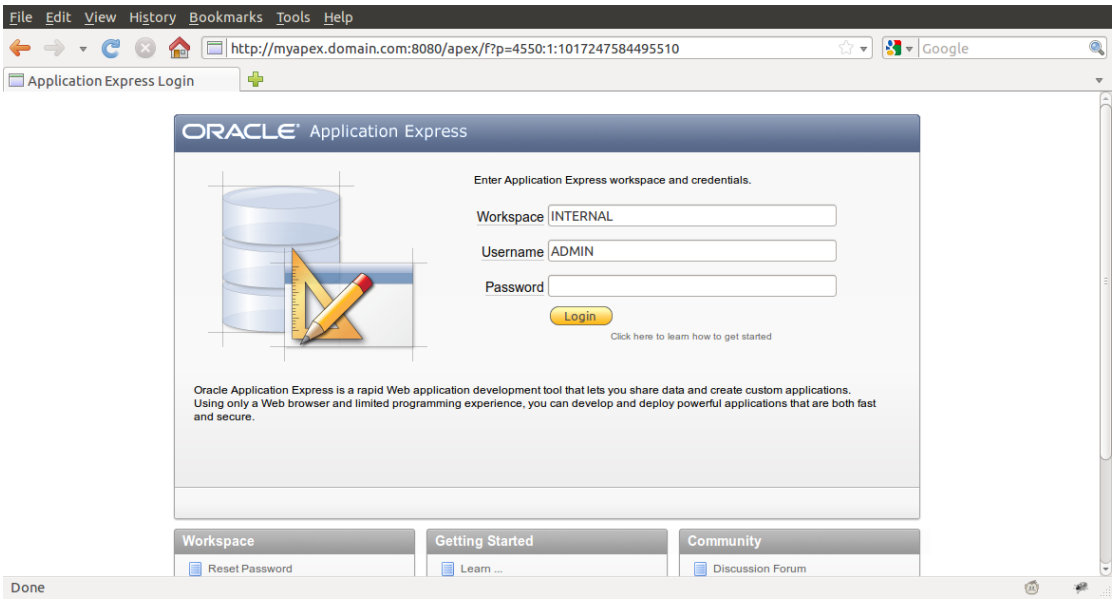

Figure 3 Apex administrative interface log-in

Web authentication and protection

Install Apache, PL/SQL Gateway, CAS module

To achieve our use case, we will not use the HTTP listener that is embedded with the Oracle Database Express. We will use a standard Apache web server instead, with the Oracle open source PL/SQL gateway module and the CAS authentication client module.

The first step is to install the Apache HTTP server itself in the same server where the Oracle Database Express was installed:

```
# apt-get install apache2-mpm-worker
```

Next, we need to install the development environment in order to compile Oracle's PL/SQL gateway, i.e. the compiler, library tools and apache headers with the following command:

```
# apt-get install gcc libtool apache2-threaded-dev
```

Oracle's open source PL/SQL gateway for Apache is called *mod_owa* and is available for download from the following web site:

http://oss.oracle.com/projects/mod_owa/files/

You need to download the file, usually called *unix_all.tgz*, and transfer it to the server. Then unpack the sources in a directory of your choice:

```
# tar xzvf unix_all.tgz
```

An important note if you are going to compile in a 32-bit system. There are 64-bit objects in the archive: if you try to compile the sources, the make command will not go through compilation as it will found object files. The best thing to do is to remove all object (*.o) files if you're going to compile for a 32-bist system.

Before going through the actual compilation, a small modification to the *modowa/apache22/modowa.mk* makefile needs to be made: please ensure that the the following includes statement is in the beginning of the makefile:

```
INCLUDES = -I. $(ORAINC) -I$(APACHE_TOP)/include -
I/usr/include/apache2/ -I/usr/include/apr-1.0/
```

It's now time to compile and install as follows:

```
# cd modowa/src
# make -f ../apache22/modowa.mk
# cp mod_owa.so /usr/lib/apache2/modules/
```

The next step is to create an Apache configuration file that is needed to load the module and activate it:

```
# echo "LoadModule owa_module /usr/lib/apache2/modules/mod_owa.so" >
/etc/apache2/mods-available/mod_owa.load
# a2enmod mod_owa
```

The module is linked against the Oracle libraries, therefore Apache needs to know where the libraries are. The system library search path should be modified to include the above libraries. Ubuntu, as well as most Linux distributions, are flexible enough to create a separate file to be included in the */etc/ld.so.conf* file, therefore create a new file as */etc/ld.so.conf.d/oracle.conf* and populate it with the following directory:

/usr/lib/oracle/xe/app/oracle/product/10.2.0/server/lib/

Next, re-create the library cache by using the *ldconfig* command as root. Now Apache can be safely restarted, ensure that no errors are present in the apache log files. Do not configure an Apache website at this time, as you will need to configure CAS at the same time. The CAS authentication module for apache is available as an Ubuntu package, so it can be easily installed as follows:

```
# apt-get install -y libapache2-mod-auth-cas
```

Then enable the module as follows:

```
# a2enmod auth_cas
```

Configure the authentication module

The first configuration that needs to be done is the basic CAS configuration, i.e. pointers to the CAS authentications server and the root certificate. Modify */etc/apache2/mods-available/auth_cas.conf* with the following:

```
CASloginURL https://cas-server.mydomain.com/cas/login
CASValidateURL https://cas-server.mydomain.com/cas/serviceValidate
CASCertificatePath /usr/share/ca-certificates/mycert.crt
```

The *CASloginURL* and *CASValidateURL* refer to the actual CAS hostname plus optional port, while the *CASCertificatePath* must refer to the X.509 certificate (in PAM format) of the Certification Authority that signed the certificate of the CAS. This parameter should be populated also in case of a self-signed certificate.

Configure the PL/SQL Gateway

The last step before creating the web site in apache is setting a password for APEX_PUBLIC_USER: this user is used internally by the Oracle embedded PL/SQL gateway, but it has to be unlocked and its password changed before it's used into the apache configuration file. The following sqlplus statements will change the password and unlock the user:

```
SQL> alter user APEX_PUBLIC_USER identified by password;
SQL> alter user APEX_PUBLIC_USER account unlock;
```

Replace *password* in the first statement with the password of your choice: it might be a good idea to create a random password. Now it's time to create the web site configuration file: for example, create a file */etc/apache2/sites-available/oracle* and populate it with the following configuration directives:

```
<VirtualHost 192.168.1.1:80>
      ServerName myapex.domain.com
      Alias /i/ /usr/local/apex/images/

      <Location /apex>
            AuthType CAS
            Require valid-user
            Options        None
            SetHandler     owa_handler
            OwaEnv            ORACLE_HOME
/usr/lib/oracle/xe/app/oracle/product/10.2.0/server
            OwaUserid      APEX_PUBLIC_USER/password@XE
```

```
            OwaDiag         SQL MEMORY ERROR
            OwaDocProc      "wwv_flow_file_mgr.process_download"
            OwaPool         10
            OwaStart        "f"
            order           deny,allow
            allow           from all
      </Location>
</VirtualHost>
```

In the configuration file above please note that:

1. The "/i/" image directory has been aliased to its correspondent filesystem directory to have faster images loading time.
2. Authentication has been set to CAS and it requires a valid user. A good idea might be restricting the users to a given set.
3. The ORACLE_HOME environment variable has been set via the *OraEnv* directive: although set in the environment, apache is being started from the boot scripts, therefore the ORACLE_HOME will not be effective. Changing the Apache startup script is not a good idea, as future Apache upgrades might change the startup script, making the PL/SQL gateway ineffective.
4. The directive *OwaUserid* has been set to the APEX_PUBLIC_USER and the password as set in the steps above.
5. The directive *OwaStart* points to the stored function "*f*", which is the one that handles the dispatch to the Apex application.

Once the file has been saved, enable the web site with the following command:

```
# a2ensite oracle
```

Now you can try to access your Apex application through the new stack, for example by using the following URL:

http://myapex.domain.com/apex/f?p=100:1

In the above URL, 100 is the Apex application ID. In case you are using the sample application, you should use 100 as application ID. If everything worked correctly, you should be redirected to the CAS authentication server and asked for login, then back to your Apex web application.

Custom authentication scheme

At this stage, the user is prompted for a username and password or a transparent smartcard login via CAS login page. However, the Apex application is not aware yet of which user is actually logged in and which privileges to apply. Oracle

Application Express comes with standard authentication schemas (ex: database user, apex user, ...), but it let you create your own schema to authenticate the user.

We will now use a custom authentication schema to gather user information: Apache CAS module will populate the REMOTE_USER variable. We will get this variable within a PL/SQL function, create the apex session and associate the username in the session. We will name the function *CAS_PAGE_SENTRY*.

Use a SQL tool to connect to the XE instance, for example using Oracle SQL Developer or even sqlplus. To connect to the instance, you should use the workspace name as username and the password you should have defined in the workspace creation process. Once in the tool, we will create the function using the PL/SQL source code described in *Appendix A*. Once the function has been created and compiled, we will go back to the workspace administrative web interface and the following steps have to be made in order to activate this function as our authentication schema:

1. Open your applications shared components and click on *Authentication Schemes*.
2. Click on *Create* and then select *From Scratch*. Press the *Next* button.
3. Now give a name and describe your authentication schema, for example "CAS_SCHEMA" and press *Next.*
4. Go down the page until you fin the "Page Sentry Function.." text box and populate it with "RETURN CAS_PAGE_SENTRY;" (be aware of the final semi-colon) and click *Next*.
5. Press *Next* when prompted for the session verification function. In the page, choose the URL radio box for the value "When the page sentry function...". This is the URL is redirected when an authentication is failed. It is better to type a URL that is not protected by a page sentry, for example the company's intranet. Press *Next.*
6. Leave the pre-authentication process blank and click on the *Next* button.
7. When prompted for the credentials verification method, choose "Do not verify credentials" and press *Next*.
8. Leave blank for the post-authentication process and press *Next*.
9. Leave blank when prompted for the cookie setting and press *Next*.
10. The *Logout URL* is the URL is redirected when the logout button is pressed. It is better to type a URL that is not protected by a page sentry, for example the company's intranet.

Make this schema the default, by going to the "Change Current" tab and selecting "CAS_SCHEMA" in the "Available authentication scheme" field, press *Next* and click on the "Make Current" button.

Test that the procedure is working by connecting to the main page of the application, for example:

http://myapex.domain.com/apex/f?p=100:1

If you used the Oracle sample application, you will find the username in the upper-right portion of the screen, as showed in the Figure 4 in page 28.

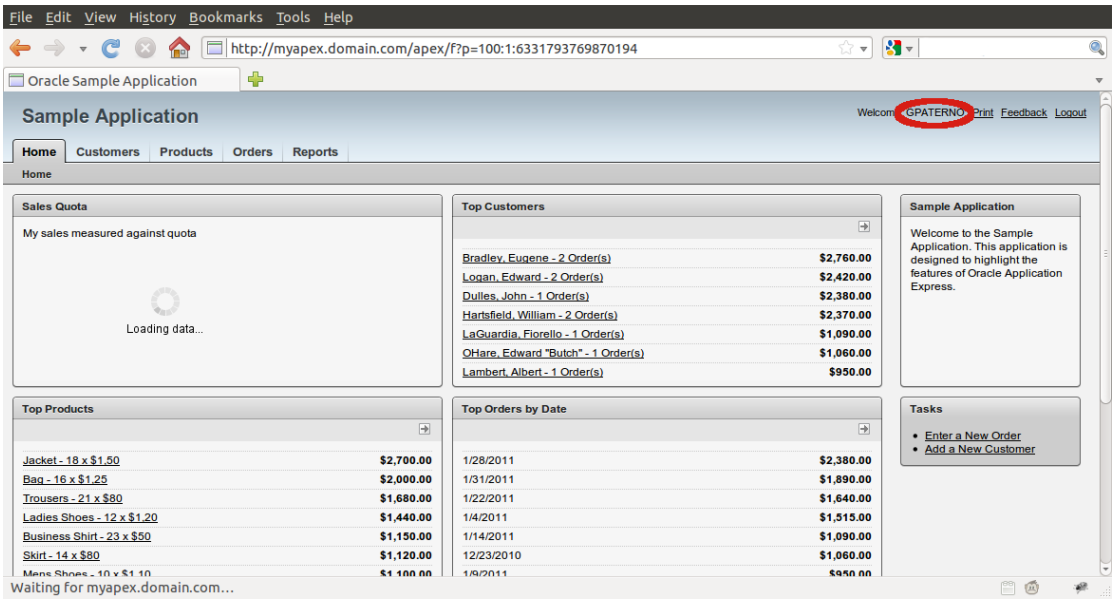

Figure 4 Logged-in user in Oracle sample

Protecting Oracle PL/SQL Gateway from abuse

There are two major issues by exposing Oracle Apex to the Internet, Intranet or Extranet. The first one is that Apex can't expose just one application, instead it will expose every application via web, including the administrative interface and the development interface (if a development installation was carried out). The second issue is that Apex in the past was prone to some vulnerabilities (ex: SQL injection, XSS, etc...) and, although Oracle is doing a great effort to make Apex more secure, it is not possible to guarantee that your application is free from security issues.

One of the way to protect Oracle against these two major threats is to leverage a Web Application Firewall (or WAF): a WAF is a specialized firewall which controls access to a web site or a web service. It basically monitors and blocks the input and output from/to a web server and/or application server if it does not meet the configured policy requirements. By using patterns, most of the WAFs on the market protect web servers from common attacks.

Before installing and configuring a Web Application Firewall, we need to understand the anatomy of an Apex request. A typical Apex URL is similar to the example below:

http://apex.domain.com:8080/apex/f?p=102:1:1774028776608804::NO

Everything after the "*/apex/*" subdirectory will be interpreted as a stored function inside an Oracle instance. The above “f” function is the stored procedure invoked, which is the most common stored procedure that handles web access, while “p” is the parameter that accepts the following string:

App:Page:Session:Request:Debug:ClearCache:itemNames:itemValues:PrinterFriendly

Let's analyse the parameters above:

1. *App* is the ID or alias of the application;
2. *Page* is the page id or alias of a given web page referred in the application; *Session* is the user session ID, that can be 0 for Public Pages or empty (in that case a new session will be created);
3. *Request* is a request keyword, it's basically a string you can specify to react in a process or region condition to control the behavior of your page;
4. *Debug* is a flag parameter that if set to YES switches on the Debug-Mode which renders debug-messages and timestamps in your browser window;
5. *ClearCache* is a page ID or a list of comma-separated page IDs to clear the cache for these pages (set session state to null, …);
6. *itemNames* is a comma-delimited list of item names used to set session state with a URL;
7. *itemValues* is a comma separated list of values. Each value is assigned to the corresponding item provided in itemNames (first value assigned to first parameter, second value assigned to second parameter, and so on);
8. *PrinterFriendly* is a flag that, if set to YES (uppercase), switches the page into PrinterFriendly-Mode, by using the Printerfriendly template to render the Page.

In the list of parameters, maybe the most dangerous one is *Debug*, that is able to show precious information that you might not want to disclose to a potential attacker. In version 4.x of Apex, is possible to disable this parameter within the application before going into production, but it might be useful to instruct our WAF to block any debug requests.

In our scenario, we will use ModSecurity, a popular open source WAF. ModSecurity is an apache module that is packaged for most of the Linux distributions, including Ubuntu. The following command will install ModSecurity in our system:

```
# apt-get install libapache-mod-security mod-security-common
```

We will use the Core Rule Set (CRS) that comes with the Ubuntu package. We suggest to periodically update your rule set by downloading the latest CRS from the official website. Also, ensure that the latest CRS is compatible with the ModSecurity module version. First of all, it is useful to create a dedicated directory for the local rules, for example:

```
# mkdir /etc/apache2/modsecurity.d
```

To use the CRS provided with the Ubuntu package, copy the examples with the following command:

```
# cp -R /usr/share/doc/mod-security-common/examples/rules/*
/etc/apache2/modsecurity.d/
```

Be aware that, as of the time of writing, there's a small bug in the rule file *base_rules/modsecurity_crs_41_phpids_filters.conf* in line 37. Disable this rule or modify as suggested in *Appendix B*.

Create */etc/apache2/mods-available/mod-securit*y.conf and populate it with the following apache configuration directives that load the ruleset:

```
<IfModule mod_security2.c>
        # This is the ModSecurity Core Rules Set.
        Include modsecurity.d/*.conf
        Include modsecurity.d/base_rules/*.conf
        Include modsecurity.d/modsecurity_localrules.conf
</IfModule>
```

The Common Rule Set comes with an example of configuration, for the sake of this paper we will copy the default configuration file:

```
# cd /etc/apache2/modsecurity.d/
# cp modsecurity_crs_10_config.conf.example
modsecurity_crs_10_config.conf
```

The steps above enable a basic configuration of ModSecurity, making it respond to standard attacks as defined in the CRS. However, we need to instruct ModSecurity to limit the access to our application exclusively and to disallow the debug flag. We need to create a file called */etc/apache2/modsecurity.d/modsecurity_localrules.conf* with the following content:

```
SecDataDir /tmp


# Allow only my application
SecRule ARGS:p "!^100:[0-9]+.*" chain,deny,log,status:404
SecRule PATH_INFO "/f"


# Protect from debug attempts (should be disabled by the app anyway)
SecRule ARGS:p "^[0-9]+:[0-9]+:[0-9]*:\w*:[Yy][Ee][Ss].*"
chain,deny,log,status:404
```

```
SecRule PATH_INFO "/f"


SecAuditEngine RelevantOnly
SecAuditLog /var/log/apache2/modsec_audit.log
```

The actual rules that protect Apex are the one that begin with *SecRule*. The first rule denies access and logs when a user tries to access the URL page "/f" (which is our stored function) with an argument *p* that is not "100", i.e. our sample application ID. Basically, if a user tries to access an Apex application with an ID different than "100", the user will receive a 404 error, i.e. not found. The number should be substituted with the application ID of your apex application, if different. The second rule is similar to the first one, except that is searches for the debug flag set to YES: if someone try to specify the debug flag, ModSecurity will return a 404 error.

With this technique, ModSecurity will protect any Apex application from common web attacks and from unauthorized use to other applications, including administrative and developer access.

Conclusions

Oracle Application Express is a simple yet powerful web application framework that can address specific rapid application needs. However, it does not have out-of-the box functionalities such as strong authentication and it showed some security vulnerabilities in the past. Through the proposed architecture, it is possible to fill these gaps by leveraging one of the most flexible web single sign-on system and web application firewall in the Open Source world, providing an highly secure environment to run your own business applications following the defense in depth philosophy.

Acknowledgments

A big thank you goes to: Pancrazio "Ezio" De Mauro, for reading and correcting my English; Mark Shuttleworth, for being so kind of providing me a preface for this publication; my wife, for being so patient while conducting tests and being with me for better and for worse; all the friends and relatives, with their help sustained me in a difficult moment.

Thanks also to the Shibboleth team, for borrowing some of the PL/SQL code. Greetings go to the member of the Italian security mailing list *sikurezza.org.*

Appendixes

Appendix A: PL/SQL Authentication Function

```
create or replace
FUNCTION                        "CAS_PAGE_SENTRY"
(p_apex_user in VARCHAR2 default 'APEX_PUBLIC_USER')
return BOOLEAN
is
      l_current_sid NUMBER;
      usr VARCHAR2(100);
      l_authenticated_username VARCHAR2(256) :=
UPPER(owa_util.get_cgi_env('REMOTE_USER'));
BEGIN
    IF l_authenticated_username IS NULL THEN
       RETURN FALSE;
    END IF;


    l_current_sid :=
wwv_flow_custom_auth_std.get_session_id_from_cookie;
    IF wwv_flow_custom_auth_std.is_session_valid THEN


        apex_application.g_instance := l_current_sid;


        IF l_authenticated_username =
wwv_flow_custom_auth_std.get_username THEN
            wwv_flow_custom_auth.define_user_session(
                p_user=>l_authenticated_username,
                p_session_id=>l_current_sid);
            RETURN TRUE;
        ELSE -- username mismatch. Unset the session cookie and
redirect back here to take other branch
            wwv_flow_custom_auth_std.logout(
                p_this_flow=>v('FLOW_ID'),

p_next_flow_page_sess=>v('FLOW_ID')||':'||NVL(v('FLOW_PAGE_ID'),0)||'
:'||l_current_sid);
            apex_application.g_unrecoverable_error := TRUE; -- tell
apex engine to quit
            RETURN FALSE;
        END IF;
```

```
    ELSE
        wwv_flow_custom_auth.define_user_session(
            p_user=>l_authenticated_username,
            p_session_id=>wwv_flow_custom_auth.get_next_session_id);
        apex_application.g_unrecoverable_error := TRUE; -- tell apex
engine to quit

        IF owa_util.get_cgi_env('REQUEST_METHOD') = 'GET'  THEN
            wwv_flow_custom_auth.remember_deep_link(p_url => 'f?'||
wwv_flow_utilities.url_decode2(owa_util.get_cgi_env('QUERY_STRING')))
;
        ELSE
           wwv_flow_custom_auth.remember_deep_link(p_url=>'f?p='||
                TO_CHAR(apex_application.g_flow_id)||':'||

TO_CHAR(NVL(apex_application.g_flow_step_id,0))||':'||
                TO_CHAR(apex_application.g_instance));
        END IF;

        wwv_flow_custom_auth_std.post_login(
            p_uname     => l_authenticated_username,
                  p_session_id =>nv('APP_SESSION'),
            p_flow_page =>
apex_application.g_flow_id||':'||NVL(apex_application.g_flow_step_id,
0));
        RETURN FALSE;
    END IF;
END CAS_PAGE_SENTRY;
```

Appendix B: Rule modification in the CRS

In the file *base_rule/modsecurity_crs_41_phpids_filters.conf* modify rule in line 37 from:

```
#SecRule
REQUEST_BODY|REQUEST_URI_RAW|ARGS|ARGS_NAMES|FILES|FILES_NAMES|XML:/*
"(?:data:.*,)|(?:\w+\s*=\W*(?!https?)\w+:)|(jar:\w+:)|(=\s*\"?\s*vbs(
?:ript)?:)|(language\s*=\s?\"?\s*vbs(?:ript)?)|on\w+\s*=\*\w+\-\"?"
"phase:2,capture,multiMatch,t:none,t:urlDecodeUni,t:urlDecodeUni,t:ht
mlEntityDecode,t:replaceComments,t:compressWhiteSpace,t:lowercase,ctl
:auditLogParts=+E,block,nolog,auditlog,msg:'Detects data: URL
injections, VBS injections and common URI schemes',id:'phpids-
27',tag:'WEB_ATTACK',logdata:'%{TX.0}',severity:'2',setvar:'tx.msg=%{
rule.msg}',setvar:tx.anomaly_score=+20,setvar:tx.%{rule.id}-
WEB_ATTACK-%{matched_var_name}=%{matched_var}"
```

To the following rule:

```
SecRule
REQUEST_BODY|REQUEST_URI_RAW|ARGS|ARGS_NAMES|FILES|FILES_NAMES|XML:/*
"(?:data:.*,)|(?:\w+\s*=\W*(!https?)\w+:)|(jar:\w+:)|(=\s*\"?\s*vbs(?
:ript)?:)|(language\s*=\s?\"?\s*vbs(?:ript)?)|on\w+\s*=\*\w+\-\"?"
"phase:2,capture,multiMatch,t:none,t:urlDecodeUni,t:urlDecodeUni,t:ht
mlEntityDecode,t:replaceComments,t:compressWhiteSpace,t:lowercase,ctl
:auditLogParts=+E,block,nolog,auditlog,msg:'Detects data: URL
injections, VBS injections and common URI schemes',id:'phpids-
27',tag:'WEB_ATTACK',logdata:'%{TX.0}',severity:'2',setvar:'tx.msg=%{
rule.msg}',setvar:tx.anomaly_score=+20,setvar:tx.%{rule.id}-
WEB_ATTACK-%{matched_var_name}=%{matched_var}"
```

Appendix C: Creative common license

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

"Adaptation" means a work based upon the Work, or upon the Work and other pre-existing works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered an Adaptation for the purpose of this License.

"Collection" means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined above) for the purposes of this License.

"Distribute" means to make available to the public the original and copies of the Work through sale or other transfer of ownership.

"Licensor" means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.

"Original Author" means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the

person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.

"Work" means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.

"You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

"Publicly Perform" means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.

"Reproduce" means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

2. Fair Dealing Rights. Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

1. to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections; and,
2. to Distribute and Publicly Perform the Work including as incorporated in Collections.

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats, but otherwise you have no rights to make Adaptations. Subject to 8(f), all rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Section 4(d).

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

a. You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Collection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(c), as requested.
b. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.
c. If You Distribute, or Publicly Perform the Work or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable

means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work. The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Collection, at a minimum such credit will appear, if a credit for all contributing authors of Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

For the avoidance of doubt:

Non-waivable Compulsory License Schemes. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;

Waivable Compulsory License Schemes. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License if Your exercise of such rights is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(b) and otherwise waives the right to collect royalties through any statutory or compulsory licensing scheme; and,

Voluntary License Schemes. The Licensor reserves the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License that is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(b).

Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION,

WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
b. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
c. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
d. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

e. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.

Page intentionally
Left blank

About the author

Giuseppe "Gippa" Paternò is an expert in computer security and complex datacenters architectures. He is CCNP, RHCSS, RHCA and RCDS certified, member of the Italian Linux Society and an active member of the Italian community of sikurezza.org.

His passion drove Giuseppe to explore all areas of computing from an early age, with particular focus on security and networking, and new technological challenges.

He's currently contracted as EMEA OEM Technical Manager by Canonical in the OEM Engineering team. He worked in the past as Solution Architect and EMEA Security Expert for Red Hat, Sun Microsystems and IBM.

Strong Authentication and Security for Oracle Application Express

Oracle Application Express is a simple yet powerful web application framework that can address specific rapid application needs, from small businesses to larger enterprises. However, it is missing an out-of-the box strong authentication functionality, such as One Time Passwords (OTP) keys or smart card. Moreover, the administrative interface and all hosted applications are potentially reachable by an attacker. This publication proposes an architecture to fill these gaps, providing an highly secure environment to run your own business applications.