

# **BlitzLocker**

A Login Manager for Salesforce

Charlie McConnell

cmcconne@mines.edu

Jack Rosenthal

jrosenth@mines.edu

Jacob Thompson

jacthomp@mines.edu

2017-06-20

# 1 Introduction

Presumably, Salesforce developers use a number of development sites:

1. mobile1 and mobile2, which track the latest development
2. blitz01 through blitz04, which track major release versions on a rotational basis
3. Pre-release versions sent to clients for application upgrades
4. Production sites

In addition, developers may be accessing sites on their own machine (typically located at localhost with a port number).

For all of these sites, the developers may have one or even multiple accounts to keep track of, creating a challenge for login management. Ordinary tools like LastPass do not keep track of different passwords per subdomain correctly, and provide no easy way of opening a logged in site in a private browsing window.

BlitzLocker attempts to fill this hole by:

1. Providing a cross-browser and cross-platform system tray icon with convenient access to login information
2. Providing configuration to quickly add and remove Salesforce development sites
3. Providing configuration to which browser should be logged into (and optionally in private browsing mode)

## 2 Requirements

Fundamentally, the BlitzLocker application should be able to save and recall login information for Salesforce development sites. Our team has identified the following functional requirements to make this happen.

The BlitzLocker application should provide:

- Configurable site URLs
- The ability to store multiple logins per site
- An editable and optional description field for each account
- An interface to randomly generate a password
- Capability of interfacing with Google Chrome

- Capability of interfacing with Apple’s Safari browser
- Capability of interfacing with Mozilla Firefox

In addition, our team has identified the following as non-functional requirements for the application:

- Easy to install and use by both developers and project owners
- A consistent interface for the GTK+3/Linux and Mac OS X operating systems

## 2.1 Non-Requirements

It’s important to clarify what BlitzLocker is not required to do, as “password manager” may be a misleading title for our application.

- **Password Encryption:** Password encryption is not necessary, as these salesforce development passwords do not protect private data
- **Network Synchronization:** The program does not have to keep data in sync using a computer network
- **Autofill Password Fields:** Salesforce sites have means of automatic login using a query string (explained later)
- **Work on Microsoft Windows:** Salesforce developers use only Linux and Mac OS X

## 3 Software Architecture

BlitzLocker consists of three components: a database to store the login information and sites, the interface to the browser to open sites, and the user interface.

### 3.1 Database Model

The database for this application is stored as a SQLite database in the user’s home directory. The model is trivial, it contains only `Org` and `Site` entities and relations between them. This is seen in Figure 1.

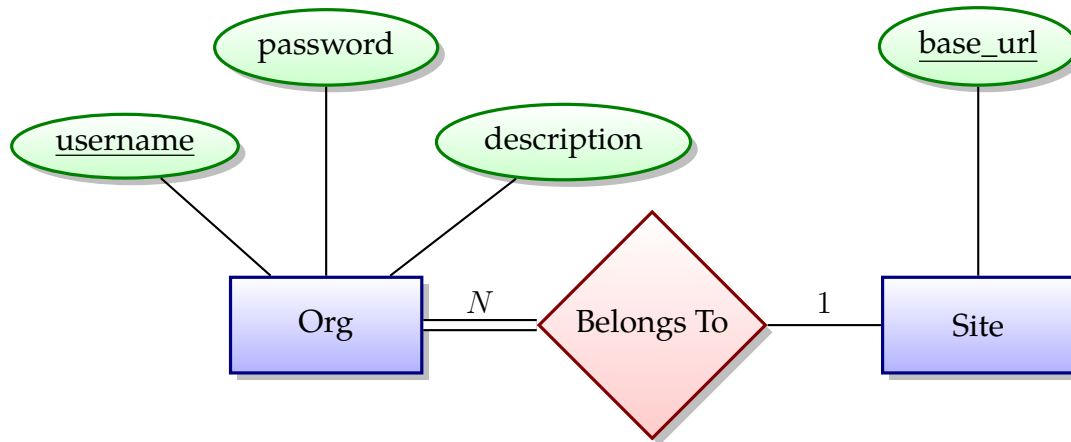


Figure 1: The Database Model

### 3.2 Browser Interface

Opening a Salesforce site in the web browser automatically logged in is made simple, as the Salesforce sites allow login information to be passed in the query string to the URL. An example is shown below:

```
https://znggcnff.salesforce.com/?un=myusername&pw=verysecret
```

In addition, most browsers have a flag to open a private browsing window. So opening Mozilla Firefox in private browsing with the specified URL is as simple as:

```
firefox --private-window https://zngg...
```

### 3.3 User Interface

The user interface is implemented using the Python bindings for GTK+ 3. This is to ensure maximum interoperability between Linux and Mac OS X.

When the system tray icon is clicked, a menu similar to the one in Figure 2 is shown.



Figure 2: The System Tray Dropdown Menu

Clicking "Login..." opens the dialog shown in Figure 3. When "OK" is clicked, the configured browser opens the site.

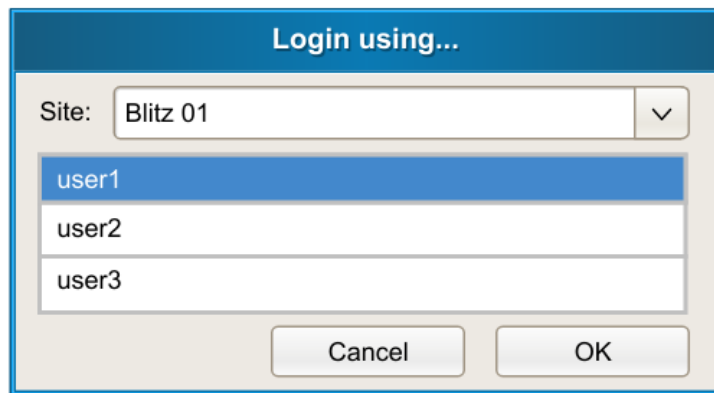
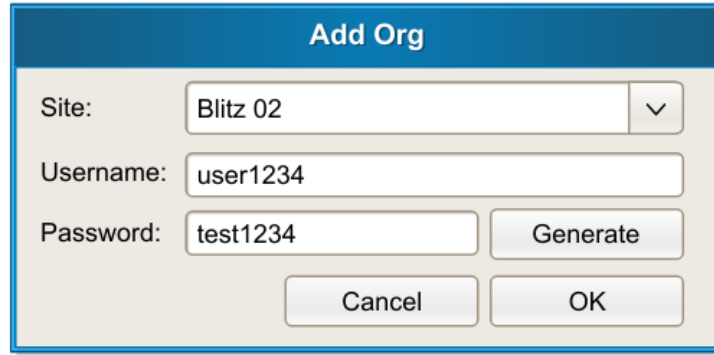


Figure 3: The Login Dialog

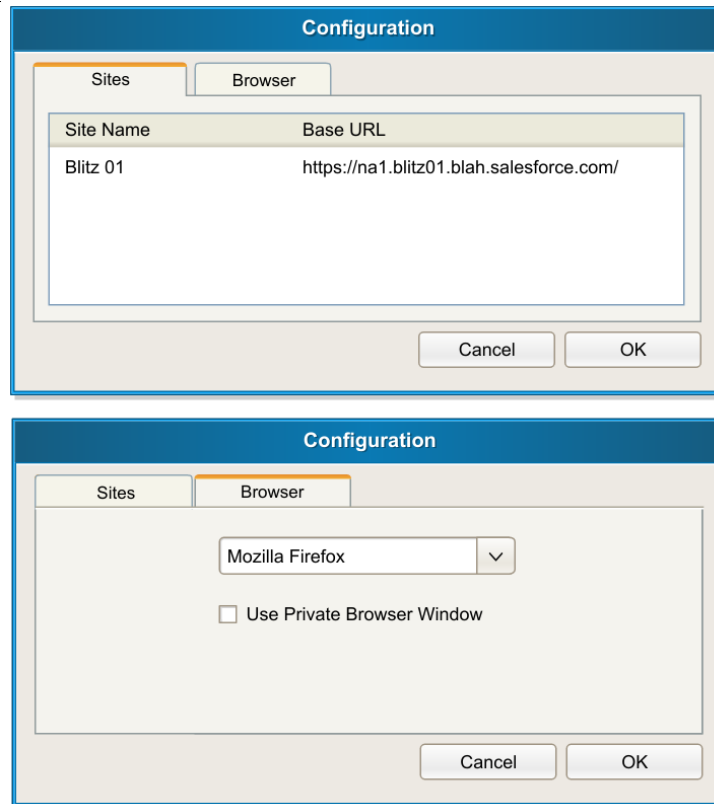
When "Add Org..." is clicked, the dialog shown in Figure 4 is opened.



The 'Add Org' dialog box features a blue header with the title 'Add Org'. It contains three input fields: 'Site:' with a dropdown menu showing 'Blitz 02', 'Username:' with the text 'user1234', and 'Password:' with the text 'test1234'. A 'Generate' button is positioned to the right of the password field. At the bottom, there are 'Cancel' and 'OK' buttons.

Figure 4: Adding an Org

The configuration dialog, as seen in Figure 5, is used to configure the default browser and sites.



The 'Configuration' dialog box is shown in two states. The top state has the 'Sites' tab selected, displaying a table with the following data:

Site Name	Base URL
Blitz 01	https://na1.blitz01.blah.salesforce.com/

The bottom state has the 'Browser' tab selected, showing a dropdown menu with 'Mozilla Firefox' and a checkbox labeled 'Use Private Browser Window' which is currently unchecked. Both states include 'Cancel' and 'OK' buttons at the bottom.

Figure 5: The Configuration Dialog

## 4 Technical Design

Since BlitzLocker must not only be easy to use, but easy to install as well, our team had to design a way that the application could be distributed with its dependencies, including the Python interpreter and GTK+ 3.

To do so, we used PyInstaller. On the Macintosh, PyInstaller is capable of bundling a Python application, all of its dependencies, and the Python interpreter into a single .app folder for Mac OS X. This makes installation as easy as dragging the application to the Applications folder, which is a common installation method on Mac OS X. Similarly, for Linux, this allowed us to publish the entire program and its dependencies in one executable file.

We did make a few optimizations in this process, as obviously we want the file we are distributing to be as small possible. First off, we rendered all GDK Pixbufs and used them instead of the PNGs we used during development. This allowed us to remove all GDK Pixbuf loaders from the binary. Secondly, we removed all the database interfaces we were not using in SQLAlchemy, leaving only SQLite. Between these two optimizations, we got the binary file size to less than twenty megabytes.

Experienced users may wish to carry around the Python source code instead of the binary, which checks in at less than 100 kilobytes; small enough to easily fit on a double density floppy disk.

## 5 Design Decisions

### 5.1 Programming Language

Our team's choice to use the Python programming language for all components of BlitzLocker was not only suggested by our client, but also justified technologically. Python not only emphasizes readability to allow for further extension of our program, but makes our program easier to deploy in a variety of environments by providing cross-platform support.

### 5.2 User Interface

The user interface is implemented using the Python bindings for GTK+. This allows for a cross-platform user interface that blends in with native UI controls on many systems.

Further, GTK+ is a GUI toolkit that is used with a wide variety of other applications, so it will be easy for someone else to modify our code later.

## 5.3 Database

Since the database must be stored locally on the end user's system (as per request of the client), we chose to use the SQLite database system. Using this system, the end user will not have to have any services running (such as MySQL or PostgreSQL) to support the application.

Further, Python's bindings for SQLite are in its standard library.

## 6 Results

Our team implemented the BlitzLocker application to the full specification described previously, closely following our design specifications. Screenshots of the resultant application can be found in Appendix A. In our implementation:

- The application is capable of launching the three key web browsers stated by the client (Google Chrome, Safari, and Mozilla Firefox) with a login URL, as well as additional browsers (Epiphany, Midori, and the default browser on Linux systems using `xdg-open`). The login dialog is shown in Figure 9.
- The application is capable of letting the user select private browsing tabs as the default for Chrome, Firefox, Epiphany, and Midori. This configuration is shown in Figure 13.
- The application is capable of letting the user configure multiple Salesforce development sites, as shown in Figure 12.
- The application is capable of storing usernames and passwords with an optional description, as shown in Figures 10 and 11.
- The application was tested on many of the commonly used Linux desktop environments (GNOME, Cinnamon, KDE, Unity, and i3) and only one incompatibility was found (explained further below). The application is shown running in GNOME in Figure 6.
- The application was tested in Mac OS X 10.9 and no incompatibilities were found (although we did have to make one modification, explained below).

### 6.1 Incompatibilities

- By no fault of our own, the Unity desktop environment requires users to authorize applications to place icons in the system tray. Since there is no way for us to automate this (by the design of Unity), users will have to enable this setting for our application to work with Unity.



- Safari provides no means of opening a new private browsing tab automatically, so the private browsing option is not available when the user selects Safari.
- There was an odd bug in the version of GTK+ 3 we were using preventing the status icon from opening a menu on some versions of Mac OS X. As a result, we used an alternative (yet very similar) interface by embedding the BlitzLocker menu in the application menu and adding an icon to the dock. This is shown in Figures 7 and 8 (in Appendix A).

## 6.2 Lessons Learned

- GTK+ 3 is a very powerful GUI toolkit with many widgets to choose from to design a graphical interface.
- Mac OS X will behave differently from what was assumed developing on Linux.
- SQLAlchemy provides means to simplify database access in Python code that is simple to understand with basic database knowledge.

# Appendix A: Application Screenshots

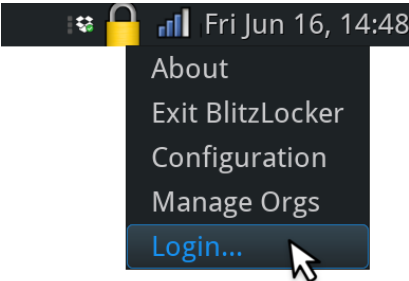


Figure 6: The Menu in GNOME 2



Figure 7: The Menu in Mac OS X



Figure 8: Dock Integration in Mac OS X

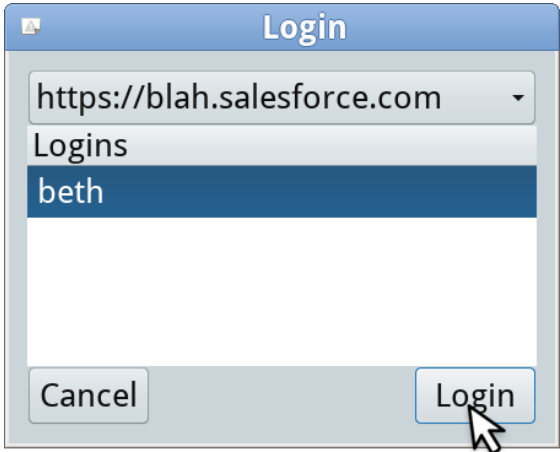


Figure 9: The Login Dialog

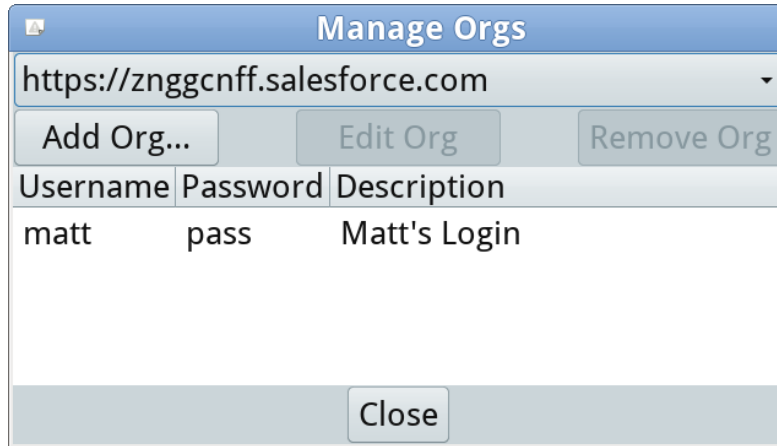


Figure 10: The Manage Orgs Dialog

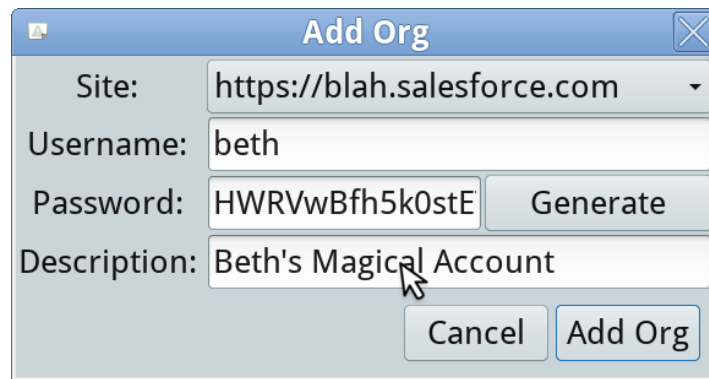


Figure 11: Adding an Org

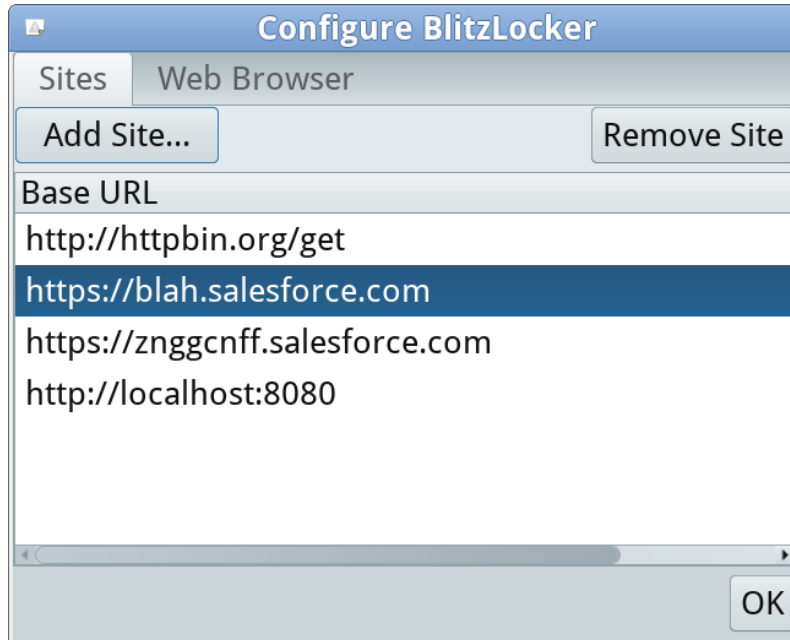


Figure 12: Configuring Sites

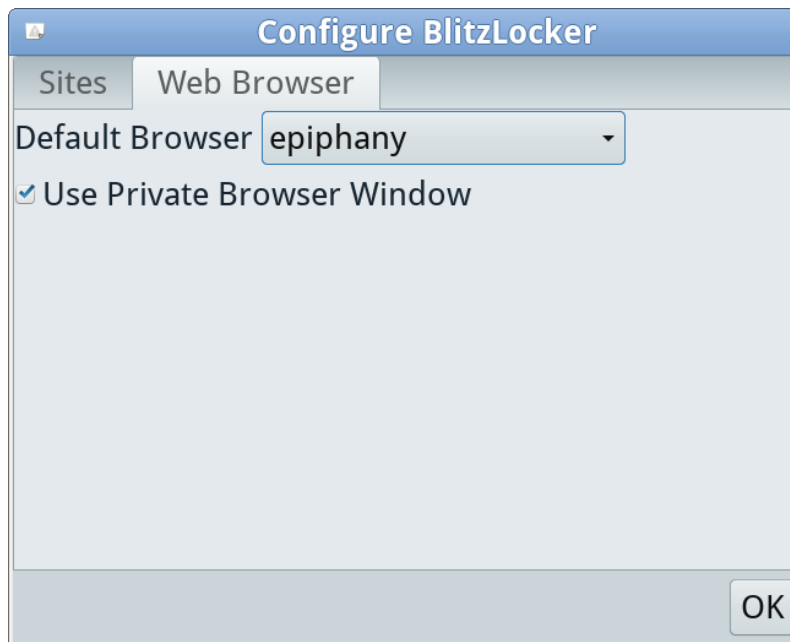


Figure 13: Configuring Default Browser