Gotham

Remote Logins Monitoring System

20 June 2016 – 21 August 2016

Author:
Mrinal Dhar

Supervisor(s):
Vincent Brillault

CERN openlab Summer Student Report 2016

Project Specification

In order to detect abused credentials, CERN is running a remote login monitoring system, called **Gotham**.

This systems compares, for each user, the location of remote logins with the user's past behaviour, notifying them of any new location. Unfortunately, the design and code used by this system is outdated and requires a complete rewrite.

The requirements of this projects are:

- Build a system with the same features as the existing one, but without any dependency on old CERN libraries (e.g. perl-LC), which would include:
    - Pulling data from a login database (running an hourly cron-job)
    - Enriching the data with geolocation and domains
    - Support for whitelisting, in particular for CERN IPs
    - Maintaining a 'known location' database
- Build a Command Line Interface (CLI) for administrator to manually list or remove locations for users
- Add support for IPv6 (currently unsupported)
- Design a new system running in real-time streaming mode (instead of using an hourly cron-job) by running the code in an Apache Spark (http://spark.apache.org/) cluster and pulling data from Apache Kafka (http://kafka.apache.org/). Special care should be taken to ensure that no data is lost in case of crashes.

In addition, extensions of this project can be considered:

- A SSO-enabled web front-end, allowing CERN users (and the CERN Computer Security Team) to review their known login locations.
- Reviewing the current location definition and evaluate alternatives. For example using 'ISPs' instead of 'Organisations', using 'City' geolocalization, etc

Abstract

This project aims to completely rewrite the Gotham Remote Logins Monitoring System currently in use at CERN. The existing system has been written in Perl, and it makes use of some really old CERN libraries that make the system difficult to maintain.

Python is a modern, widely used, high-level, interpreted programming language and, as a result, was chosen as the programming language for this project. There are a number of well-maintained open source libraries in Python that have been used for the purposes of this project,drastically decreasing the chances of security flaws in the libraries and thus simplifying the project maintenance.

Apart from the equivalent functionality that was achieved with respect to the earlier version of Gotham, a number of new features have been added, like real-time processing of input login streams, a web based frontend to be integrated with the central account management page at CERN, a REST API for accessing previous login information by other applications.

Table of Contents

1. Introduction

Computer and network security is critical to an organization like CERN. With a huge network of connected devices, it is necessary that accesses to the resources shared between these devices is authorized properly.

A significant number of CERN users log in to use these resources from outside the CERN internal network. However, there exists a possibility that some of these logins do not come from owners of those accounts, but instead from criminals who have stolen some credentials from their legitimate owners.

Currently, CERN uses a remote login monitoring system called Gotham that allows the Computer Security Team to analyse these logins and alert the affected users.

Gotham was originally written in Perl, which has lost its popularity over the years to languages like Python. This project involved re-writing Gotham in Python and adding new features. It was necessary to do so because the previous version was outdated, difficult to maintain and parts of it didn't work as intended.

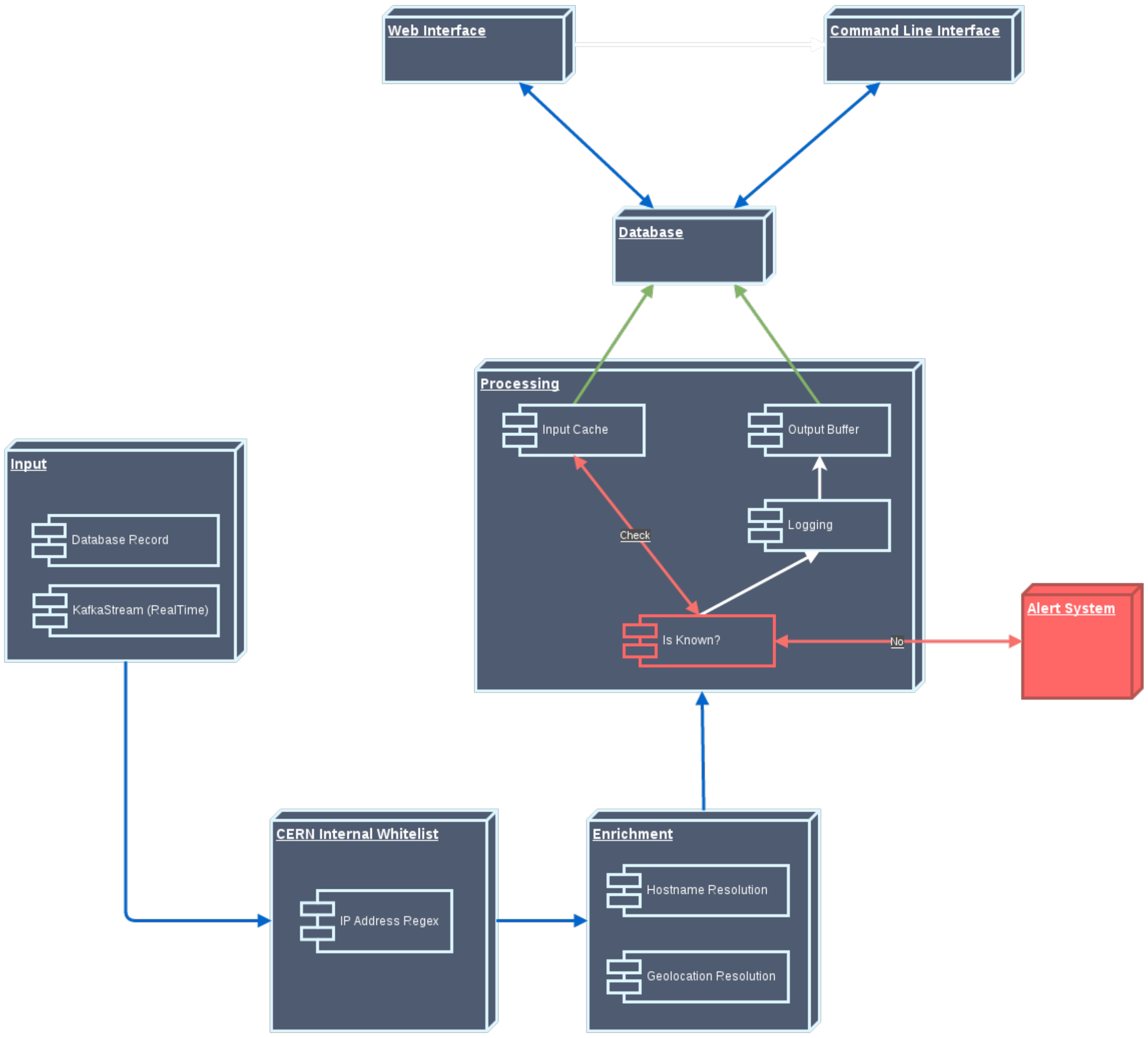
Once all the functionality of the existing system were replicated by the new implementation, which was successfully identifying logins as previously seen or otherwise, parts of the project were improved:

- Gotham's process is now distributed over a cluster of servers, by using Apache Spark, instead of running on a single system;
- The original database fetching of the input login data has been replaced by a real time streaming of input data by using Apache Kafka;
- Support for iPv6 was added.

Extensions possible to the project were also completed:

- A web based frontend written in ASP.NET, which should be directly integrated into the SSO-enabled account management portal;
- A flask (http://flask.pocoo.org/) based REST-API written in python and deployed in the bleeding-edge openshift (https://www.openshift.com/) infrastructure at CERN.

2. Workflow



Gotham's core functionality has been divided into several modules, following a pipeline manner of execution. This modular approach simplifies the debugging of the application, as each module can be understood, tested and validated individually. In addition, modules can easily be added or replaced by different implementations. The single pipeline processing through all the modules keeps the whole system simple and ensures that any new developer or maintainer can easily understand the model and the order of execution of all the modules.

First, the system needs to acquire input data that need to be processed. Two input modules exists: One, like in the old system, periodically fetches data from the CERN Computer Security Team’s login database. The other uses Apache Kafka to receive streamed real time input data that can be easily processed in Apache Kafka.

This data is first passed to the IP whitelisting module, which checks whether the login was made from inside CERN. If it does come from CERN, it is ignored by the rest of the system. Otherwise, as it comes from a remote computer, the data is passed to the next module in the pipeline, called “Enrichment”.

The Enrichment module provides valuable information about the location of the remote client based on its IP address. It has two sub-modules, the geolocation part which identifies the country and ISP name, and the DNS part, which identifies the domain name of the IP address used to make the login.

Next is the “Judgement” module, which fetches information about previous logins made by the user from an Oracle database. Using this data, it classifies new logins as “known” or “unknown”.

Based on the judgement made by the previous module, there are two ways to go forward. If the login was classified as a “known” login, there is nothing left for Gotham to do. But if the login was deemed to be “unknown”, Gotham raises an alert which will lead to an email being sent to the concerned user notifying him/her about a possible unauthorized use of their user account.

Then, the database is updated with this login, for future use by the system.

Upon receiving the alert, the user can verify whether they know about this login, and if not, choose to send an email to the Computer Security Team about this incident.

There are two ways of interacting with this work-flow:

- The CERN Computer Security Team can interact with the database by using a command line interface that is integrated with Gotham. They can list and delete previous login locations for any users.

- The users themselves can interact with Gotham's database using a web based front-end that connects to a REST-API which interfaces with the database.

  When the integration is completed, they will be able via the Account Management service (http://account.cern.ch), to list the locations that appears with their previous logins and re-enable notification for any of them.

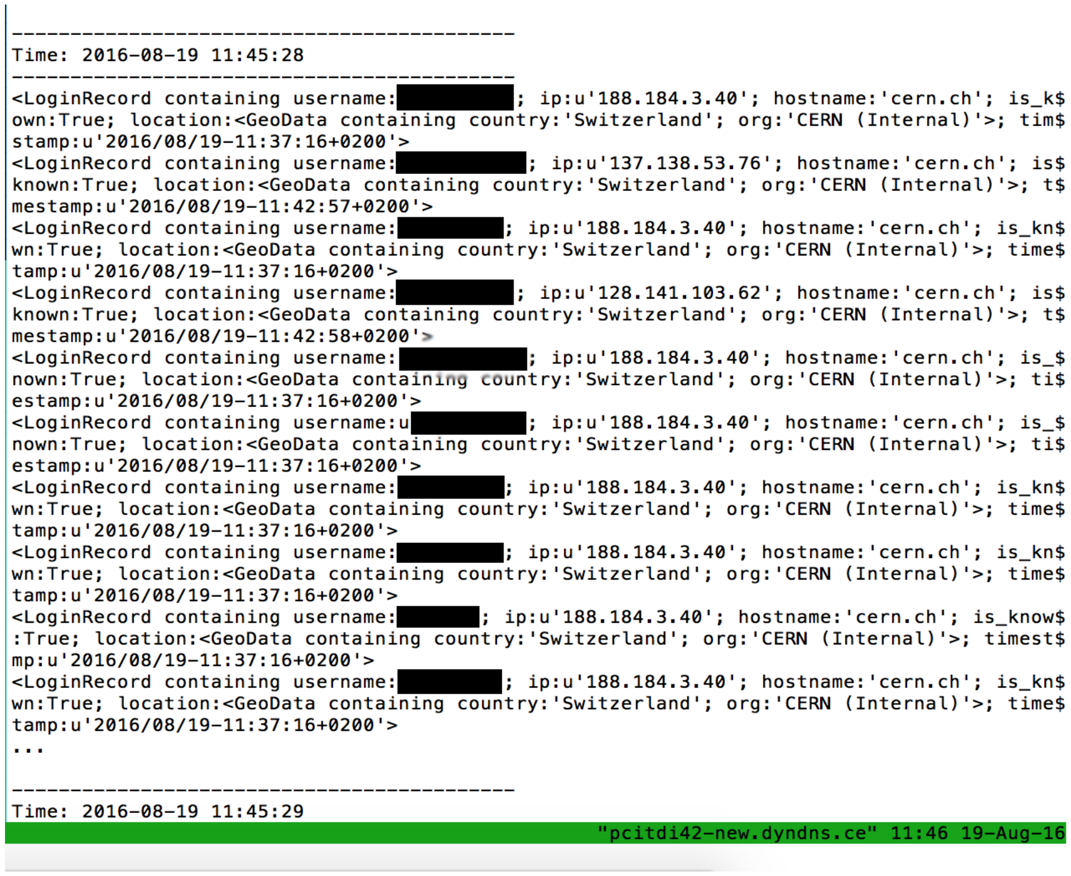
3. Input module

Two independent input methods have been implemented: one allows Gotham to run in single-node while the other allows a cluster mode. With the distributed system mode, Gotham can pull input data from Kafka stream, while in single system mode, the data is fetched from database.

In an input data tuple, Gotham expects the username of the logged in account, the IP address of the remote computer from where the login was made, and the timestamp of the login. The IP address is further processed and the username is used to specify the whole process for a particular user.

3.1 Realtime

Gotham has been integrated with Apache Kafka, which allows for real-time login information to be pulled by the system and then processed. This process can be run on a standalone machine or on a cluster, using the distributive capabilities of Spark.



```
-------------------------------------------
Time: 2016-08-19 11:45:28
-------------------------------------------
<LoginRecord containing username:[redacted]; ip:u'188.184.3.40'; hostname:'cern.ch'; is_k$
own:True; location:<GeoData containing country:'Switzerland'; org:'CERN (Internal)'>; tim$
stamp:u'2016/08/19-11:37:16+0200'>
<LoginRecord containing username:[redacted]; ip:u'137.138.53.76'; hostname:'cern.ch'; is$
known:True; location:<GeoData containing country:'Switzerland'; org:'CERN (Internal)'>; t$
mestamp:u'2016/08/19-11:42:57+0200'>
<LoginRecord containing username:[redacted]; ip:u'188.184.3.40'; hostname:'cern.ch'; is_kn$
wn:True; location:<GeoData containing country:'Switzerland'; org:'CERN (Internal)'>; time$
tamp:u'2016/08/19-11:37:16+0200'>
<LoginRecord containing username:[redacted]; ip:u'128.141.103.62'; hostname:'cern.ch'; is$
known:True; location:<GeoData containing country:'Switzerland'; org:'CERN (Internal)'>; t$
mestamp:u'2016/08/19-11:42:58+0200'>
<LoginRecord containing username:[redacted]; ip:u'188.184.3.40'; hostname:'cern.ch'; is_$
nown:True; location:<GeoData containing country:'Switzerland'; org:'CERN (Internal)'>; ti$
estamp:u'2016/08/19-11:37:16+0200'>
<LoginRecord containing username:u[redacted]; ip:u'188.184.3.40'; hostname:'cern.ch'; is_$
nown:True; location:<GeoData containing country:'Switzerland'; org:'CERN (Internal)'>; ti$
estamp:u'2016/08/19-11:37:16+0200'>
<LoginRecord containing username:[redacted]; ip:u'188.184.3.40'; hostname:'cern.ch'; is_kn$
wn:True; location:<GeoData containing country:'Switzerland'; org:'CERN (Internal)'>; time$
tamp:u'2016/08/19-11:37:16+0200'>
<LoginRecord containing username:[redacted]; ip:u'188.184.3.40'; hostname:'cern.ch'; is_kn$
wn:True; location:<GeoData containing country:'Switzerland'; org:'CERN (Internal)'>; time$
tamp:u'2016/08/19-11:37:16+0200'>
<LoginRecord containing username:[redacted]; ip:u'188.184.3.40'; hostname:'cern.ch'; is_know$
:True; location:<GeoData containing country:'Switzerland'; org:'CERN (Internal)'>; timest$
mp:u'2016/08/19-11:37:16+0200'>
<LoginRecord containing username:[redacted]; ip:u'188.184.3.40'; hostname:'cern.ch'; is_kn$
wn:True; location:<GeoData containing country:'Switzerland'; org:'CERN (Internal)'>; time$
tamp:u'2016/08/19-11:37:16+0200'>
...

-------------------------------------------
Time: 2016-08-19 11:45:29
                                              "pcitdi42-new.dyndns.ce" 11:46 19-Aug-16
```

KafkaStream delivers the login values to Gotham as these logins happen on the network.

3.2 Database

The second input module allows Gotham to load input values from a database instead of real-time values. This fall-ack mechanism, similar to the old system processing, allows previous logins to be re-evaluated by Gotham after a system failure or a change in configuration.

Currently, Oracle DB (https://www.oracle.com/database) is supported as the database system since it is widely in use at CERN.

4. Processing Module

After obtaining the login tuple of (timestamp, username, IP address), Gotham begins processing it. This happens by passing the input data from module to module, in a pipeline architecture.

The various modules in the processing phase are:

4.1 CERN Internal Whitelist

There are certain IP ranges that are used by clients that are directly on the CERN network. Since the input stream does not differentiate between internal logins and remote logins, Gotham must make this distinction itself because the logins made from within the CERN network are trusted by default, as an attacker would always need to connect from outside first and due to the fact that internal authentications of some services would always generate such events.

These IP ranges are matched using regular expression patterns in the whitelist module, and only those which do not match these patterns are allowed to be further processed. In other words, logins from computers inside CERN are ignored by the rest of the system. This regex matches both IPv4 and IPv6 addresses. Since the entire project need not be re-deployed every time there is a change in the pattern for these IP addresses, this pattern is stored and retrieved from an external configuration file, which can be changed without having to dig into any code.

4.2 Enrichment

After the logins made from within CERN have been filtered out, Gotham needs to know more about each logins in order to verify if they were made from locations known for that user previously. The enrichment module provides more information about the login based on the IP address of the remote computer from where the login was made.

The first step in enrichment involves geolocation resolution. The company “Maxmind” (https://www.maxmind.com/) provides a library API and a database which allows Gotham to map IP addresses to geographical locations like city and country. Another database provides the organization name or the ISP name of the owner of the IP address, if any. These details are critical to Gotham's functioning since — in order to check whether a new login was made by the owner — it needs to compare the new location with the previous one in the database.

After geolocation, Gotham performs domain name resolution in order to cross-validate. It is also useful in cases where a valid organization name was not obtained from the GeoIP database. The domain, extracted from every domain name, can thus also be used for verifying that an account is being logged in via a different computer than usual.

4.3 Judgement

Once Gotham has all the information it needs about a login, the final step in processing involves checking if the login must be deemed known previously for that user or should an alert be raised. A cache is used for recently fetched logins to help speed things along.

If the current login's geolocation or domain name does not match any entry in the database for that particular user, then this login is said to be "unknown" and an alert is sent to the user about this. Either way, the current login information is processed by the logging module for updating the database.

5. Output Module

5.1 Database logging

The database must be updated with the processed login information so that future logins by that user are matched against the most recent login that was made for that account.

The module processes the inputs in entire partitions, rather than individually. This allows the system to reduce the number of database queries to a manageable extent.

The database schema was changed in this system. Earlier, geolocation and domain name information was stored in one single table, combined. This was unfavourable because it made it difficult to isolate logins which were unknown according to each of them individually. Thus, two tables were used, one each for the domain name and geolocation.

The entries from the previous table were migrated using a python script that separates the domain name and geolocation and inserts them into the two different tables.

5.2 Alert System

If the judgement module determines a certain login to be unknown, Gotham issues an alert via Apache Kafka which will result in an email being sent to the concerned CERN user about a possible unauthorized login and gives them information regarding the geographical location from where the login was made. The use of Apache Kafka, which will also be used by the future infrastructure of the CERN Computer Security Team will simplify the integration of these notifications in said infrastructure.

The user can then contact the Computer Security Team for escalation of the incident, in case he/she believes that his/her account was compromised. Or, with the new web based front-end in place on the account management page, he/she can directly go there and see and manage the previous logins.

6. Interfaces

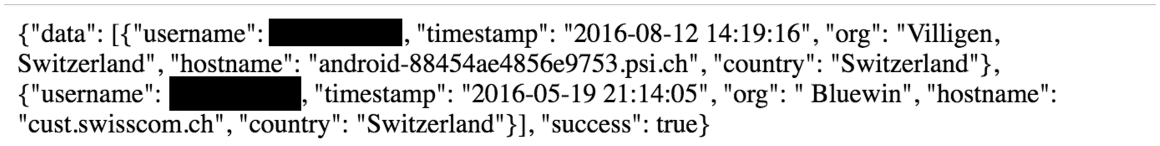
6.1 Flask-based REST API

Flask is a small, open-source framework for Python that was used as the backend for a REST API which interacts with the database for listing and deleting previous login information for users.

It supports multiple API keys, and access is granted only to those applications which use a valid API key.

The API has two endpoints :

- `/list/<type>/<value>/<api_key>`



```
{"data": [{"username": [redacted], "timestamp": "2016-08-12 14:19:16", "org": "Villigen,
Switzerland", "hostname": "android-88454ae4856e9753.psi.ch", "country": "Switzerland"},
{"username": [redacted], "timestamp": "2016-05-19 21:14:05", "org": " Bluewin", "hostname":
"cust.swisscom.ch", "country": "Switzerland"}], "success": true}
```

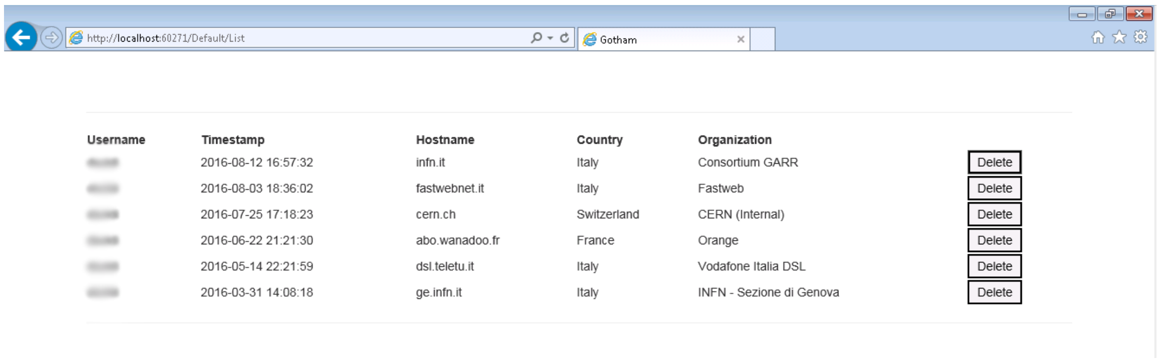
This lists previous logins based on the exact query performed.

The type attribute can take one of three values: user, country or hostname. This determines what field and table to perform the query for the value attribute on.

- `/delete/<username>/<hostname>/<country>/<org>/<api_key>`

This deletes a login entry for a user, based on the values which help it to uniquely determine which login must be deleted.

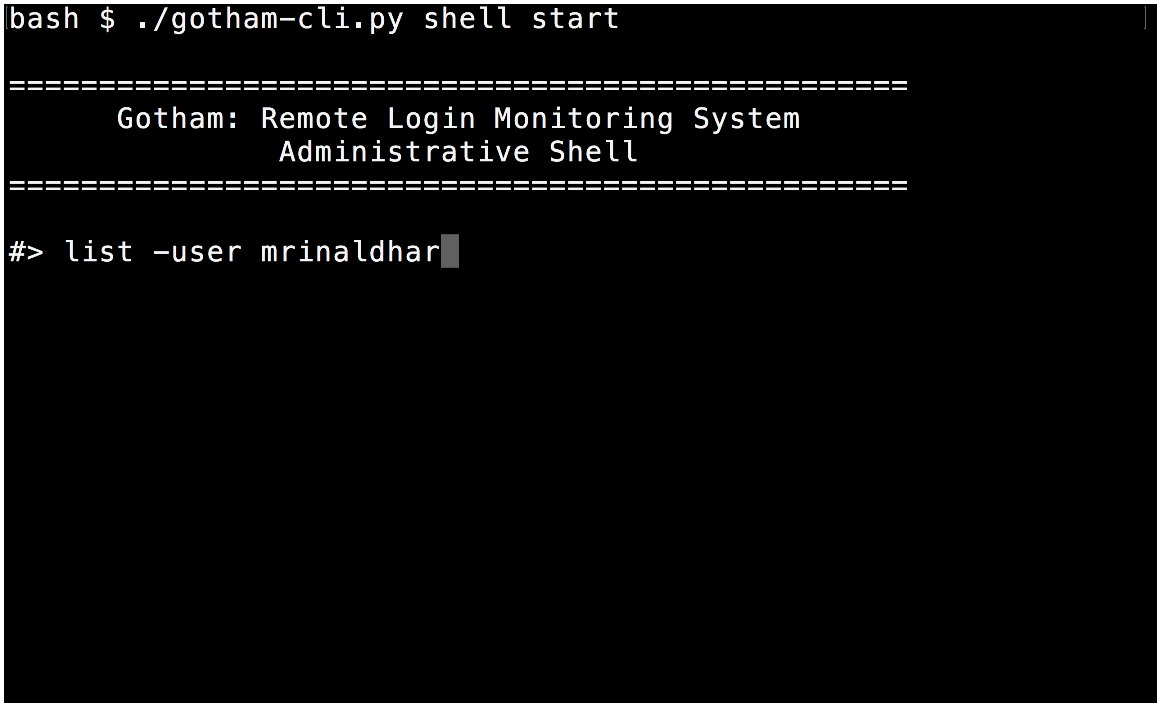
6.2 Web Front-End



Username	Timestamp	Hostname	Country	Organization	
[redacted]	2016-08-12 16:57:32	infn.it	Italy	Consortium GARR	Delete
[redacted]	2016-08-03 18:36:02	fastwebnet.it	Italy	Fastweb	Delete
[redacted]	2016-07-25 17:18:23	cern.ch	Switzerland	CERN (Internal)	Delete
[redacted]	2016-06-22 21:21:30	abo.wanadoo.fr	France	Orange	Delete
[redacted]	2016-05-14 22:21:59	dsl.teletu.it	Italy	Vodafone Italia DSL	Delete
[redacted]	2016-03-31 14:08:18	ge.infn.it	Italy	INFN - Sezione di Genova	Delete

A web based front-end has been designed, which makes use of the REST API to let the user manage their logins themselves without having to contact the Computer Security team. This has been written in ASP.NET and will be integrated with the account management page at CERN.

6.3 Command Line Interface



```
bash $ ./gotham-cli.py shell start

==================================================
      Gotham: Remote Login Monitoring System
               Administrative Shell
==================================================

#> list -user mrinaldhar
```

For use by the Computer Security Team, a command line interface has been built which supports listing and deleting previous login entries for any user. This is an administrative tool only for use by the team since it doesn't require authentication from the user to delete their login information.

7. Future Work

The next goal for this project should be integration of the web front-end with the Accounts Management Service at CERN.

There is additional work that can be done to improve the system:

- Integration of the Public Suffix List (https://publicsuffix.org/) , which provides an accurate list of domain name suffixes. This would replace the current method in use, which does not always yield the most precise result.

- The web front-end could provide even more information to the user, like old locations which had logins from but are older than three months and so are not considered by the Judgement module.