DIMS Job Descriptions Documentation

Release 2.9.1

David Dittrich

Nov 29, 2017

Contents

CHAPTER 1

Introduction

This document (version 2.9.1) contains descriptions of the basic skill sets and tasks that are involved in the Distributed Incident Management System (DIMS) project. It is based on the programming languages, operating systems, pre-existing open source components, and architectural design that make up DIMS.

As this project involves security information management and communication, general experience with computer security concepts (both attack and defense), TCP/IP networking, client/server communication using standardized protocols, and at least a basic understanding of computer forensics and computer security incident response are ideal. Where there is weakness in any of these areas, a demonstrated ability to quickly learn new technologies "on the job" will be expected.

The DIMS project is primarily a *system integration* and *software development* project that has as its deliverables a complete *functional system of systems*, with documented *test and evaluation*, *user manuals*, *software version description*, and multiple *open source software repositories* that allow someone to implement a DIMS system locally.

Skill sets described in the sections below are broken down to a level of granularity greater than the number and type of actual positions of DIMS staff, student research assistants, and contractors. In other words, these sections are not intended to map 1:1 to discrete team positions. Rather, the staff, student research assistants, and contractors working on the DIMS project may be called upon to perform tasks from more than one of these sections as needed. The exception to this statement is that *all* DIMS team members *are* expected to meet (or be willing to quickly learn) the *General Requirements* listed in the first section.

1.1 General Requirements

The DIMS project involves coordination of team members in multiple locations, multiple time zones, and multiple overlapping areas of responsibility. In order to communicate, coordinate, maintain momentum of project development, and meet deliverable requirements of the contract with the sponsor, all DIMS team members must be able to work asynchronously, independently, and be responsible for following task prioritization or asking for direction as necessary.

- Must be familiar with, or be able to quickly learn, the following skills and tools:
  - Software development using the Agile methodology and Scrum methodology
  - Use of Git and related tools for source control management (SCM)

        * Using GitHub, and *hub-flow* (hub and git-flow) following the Vincent Dreisen branching workflow
        * Handling merge conflicts from the command line with tools like `vimdiff`, or graphical user interface tools like gitk, kdiff3, or meld.
    - The concepts of Continuous Integration and DevOps (also known as *agile system administration* or *agile operations*) for rapid development, testing, and release of a functional system
    - The concepts of *test and evaluation* using *unit tests*, *functional tests*, *acceptance tests*, and *regression tests*
    - FosWiki (and its version of wiki mark-up)
- Must be able to document your work before, during, and after tasks.
    - Sphinx, ReStructured Text (reST), and ReadTheDocs are the tools used in the DIMS project
    - LaTeX experience preferred for peer-reviewed publications and posters
    - Microsoft Office products also used
- Must be able to perform independent research on tasks required for DIMS development as directed and present findings in both written form and live demonstrations (including demos via Adobe Connect.)
- Must be able to follow prioritized tasking using a Jira Agile ticketing system, and either (a) complete tasks within a 2-week development iteration (a.k.a., *sprint*) cycle, or (b) coordinate with the PI or scrum master to re-prioritize and/or break up overly-large tasks before the end of sprints.
- General understanding of networking concepts (e.g., the *client/server* model, Domain Name System (DNS) naming, the role of a Virtual Private Network (VPN) for remote access, etc.)
- Familiarity with *information security* related concepts (both *attack* and *defense*) including, *Trojan horses*, *social engineering*, *malware*, *incident response*, etc. (The kinds of things listed in General Computer Security Awareness)

1.2 Referenced Documents

The following documents describe the DIMS project and provide background material related to tasking.

- HSHQDC-13-C-B0013, "From Local to Global Awareness: A Distributed Incident Management System," Draft contract, Section C - Statement of Work (marked up version)
- "System Requirements and Concept of Operations for From Local to Gobal Awareness: A Distributed Incident Management System (DIMS)," Version 0.1, March 30, 2104.
- "Architectural Design for From Local to Global Awareness: A Distributed Incident Management System (DIMS)," Version 0.1, March 30, 2014.
- "Distributed Incident Management System (DIMS) Software System Test Plan (SSTP)," Version 1.5, May 12, 2014

1.3 Front End (User Interfaces, CLI)

There are two primary ways of accessing DIMS functionality: a *web based application* for a graphical interface, and *command line programs* that can be invoked from a command line or scripts.

1.3.1 Web application

The web application consists of a front end client application which runs in a browser and a back end server application that receives requests from the client and communicates with other DIMS programs, services and components.

Web application client

The web application client is a Javascript AngularJS application. The following skills at an intermediate to advanced level are required:

- Javascript
- AngularJS
- CSS, Less
- Bootstrap
- Promises ($q library)
- AngularJS and Javascript unit and end-to-end testing with Karma, Mocha, Chai, Jasmine, Protractor, among others
- Automation using Grunt
- Package management using npm, Bower
- JSON Web Tokens

In addition, the developer should be familiar with Javascript functional programming, general principles of web 2.0 architectures and practices.

Web application server

The backend server for the web application is written in Node.js and Express. The following skills at an intermediate to advanced level are required when working on the web application server:

- Javascript
- Node.js
- Express
- Promises (Q library- https://github.com/kriskowal/q)
- Testing using Karma, Mocha, Chai, Sinon
- Automation using Grunt
- Package management using npm
- JSON Web Tokens

In addition, the developer must be familiar with Javascript functional programming, RabbitMQ (AMQP), socket.io, Redis, Postgres.

1.3.2 Command line front end

- Java command line programs
- **Python 2.7 command line programs**

  - OpenStack's cliff - Command Line Interface Formulation Framework (GitHub openstack/cliff)

- **Bash command line programs**
  - Google's shFlags module (GitHub kward/shflags)

1.4 Back-end (Databases, Data Sources)

Programming languages and tools used:

- Python 2.7
- Java and Maven
- Perl 5.10
- Ruby 1.9.3p194
- GNU/Linux command line tools including `awk`, `sed`, `grep`, `make`, etc.
- STIX *[The12]* and other XML structured document formats
- AMQP, the Publish-subscribe model and RabbitMQ

Databases and concepts used:

- MySQL and PostgreSQL
- Redis
- The ELK stack
- Graph databases
- NoSQL databases

1.5 System and Network Administration

The DIMS project primarily uses GNU/Linux operating systems. Integrating multiple open source tools produced by others (with their choice of operating system version, libraries, etc.) requires simultaneously dealing with multiple Linux distributions and versions, where they place configuration files, how they manage service daemons, etc. Using Ansible to control all of these configuration files requires understanding how to identify and provide installation steps for each supported operating system and use of methodical development and testing practices to ensure that playbooks run on all supported operating systems.

Automation of repetitive processes through the use of `Makefile` and/or Bash scripts is used heavily in this project to facilitate build automation. Familiarity with BASH scripting and GNU/Linux command line tools (e.g, `awk`, `sed`, `grep`, `make`) is a requirement.

Experience with the following operating systems:

- CentOS 5.x & 6.x
- RedHat Enterprise Linux 5.x, 6.x
- Ubuntu 12.04 and 14.04 LTS releases
- CoreOS
- Experience with bootable Linux ISO and USB distributions (e.g., CAINE, Debian Mint)
- Use of operating system installation automation tools like Kickstart and Preseed

Experience with the following system administration concepts and tasks:

- Backup and restore
- Build automation
- Management of configuration files using a source code control system for version control, differencing between versions and across operating system distributions
- Device naming and device management in multiple Linux distributions
- Production, management, and revocation/replacement of encryption keys and certificates

Experience with the following network administration concepts and tasks:

- TCP/IP networking (routing, VPN tunneling, VLANs, `iptables` firewalls)
- Use of OpenVPN Virtual Private Network (VPN) and Virtual Local Area Network (VLAN) for network isolation
- Fundamentals of routing protocols, static routing tables, subnet allocation and subnet masks, and use of RFC 1918 non-routable address blocks as part of constructing a multi-layer Private network
- Using iptables for firewalling and Network address translation (NAT)
- Designing networks for static and dynamic host provisioning (subnet allocation, device address allocation, VPN with static and dynamic address allocation, VLAN configuration on managed switches)
- The Domain Name System (DNS), DNS recursion, "Split Horizon DNS", and dnsmasq
- Understanding of Linux network interface card (NIC) device mapping, differences in device naming across multiple Linux operating system distributions, ability to identify and document the mapping of internal logical device names to physical NIC ports to properly cable systems in a multi-switch rack

Experience with the following operating system level tools:

- Ansible
- Vagrant
- Packer
- Virtualization and hypervisors, including Virtualbox
- Docker

Experience and/or ability to install, configure, and maintain the following server software packages:

- Unix `syslog` and `rsyslog`
- The ELK stack
- MySQL Ver 14.12 and above
- PostgreSQL 8.4.17 and above
- The Collective Intelligence Framework (CIF) v1.0 and above
- SiLKTools v1.1.3
- flowtools v0.68
- HTML and CGI using Apache and Nginx
- OpenVPN
- RabbitMQ
- Botnets v0.95 (customized)

Experience with rack-mounted hardware:

- Planning rack layout
- Planning network cabling between layered switches/VLANs and to NICs in servers and orderly cable management

1.6 Project Management

- Able to manage daily *scrum* meetings and keep notes of progress on tasks in the current sprint.
- Able to keep team members on track following prioritized tasking in Jira Agile set by the PI.
- Able to solicit input and/or feedback from team members, and receive financial reporting from administative staff, in order to compile monthly status reports that are to be delivered to the sponsor no later than the last business day prior to the 15th of each month.
- Able to support production and review of documentation deliverables per contract with the sponsor.
- Able to maintain situational awareness for the team as to upcoming contractual deliverables and deadlines related to software releases.
- Able to manage test and evaluation processes related to production of Test Plan and Test Report deliverables.

1.7 Contact

Section author: Dave Dittrich (@davedittrich) <dittrich @ u.washington.edu>

1.8 License


```
Berkeley Three Clause License
=============================

Copyright (c) 2014-2017 University of Washington. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this
list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors
may be used to endorse or promote products derived from this software without
specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
```

```
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

Bibliography

[The12] The Mitre Corporation. Standarizing Cyber Threat Intelligence Information with the Structured Threat Information eXpression (STIX). http://makingsecuritymeasurable.mitre.org/docs/STIX-Whitepaper.pdf, 2012.