DIMS Commercialization and Open Source Licensing Plan

Release 1.7.0

David Dittrich

May 07, 2017

Contents

This document (version 1.7.0) describes the Commercialization and Open Source Licensing Plan for the DIMS project (DHS Contract HSHQDC-13-C-B0013, referred to in this document as "the Contract").

CHAPTER 1

Introduction

The Verizon Data Breach Investigation Report *[DBIR16]*, one of the most highly-cited sources of information on data breach causes, reports that almost 40% of intrusions are targeted at servers (p. 9), 63% of breaches are due to weak, default, or stolen password (p. 20), and about 1/3 of all breaches are due to mistakes and misconfiguration, privilege misuse, and other factors related to computing *infrastructure* apart from vulnerabilities in software (p. 22). In December 2015, Forbes estimated that the market for cybersecurity products and services in 2015 was $70 Billion and would reach$170 Billion by the year 2020. *[Mor15]*

Open source software makes up the foundation of the Internet as we know it today. OpenSSH, OpenSSL, NTP, and GnuPG are examples of successful open source software used in many products, both open source and commercial. Vulnerabilities like Heartbleed (CVE-2014-0160) have served as a wake-up call for the need to improve the development and maintenance processes of these open source projects to create a more secure foundation. The Linux Foundation Core Infrastructure Initiative aims to "speed the pace of open-source innovation while dramatically reducing global threats to online security." They do this with the Badge Program using detailed badging criteria establishing an "open source secure development maturity model."

But what about the operating systems, libraries, and configuration of services that make up the *infrastructure* within which the Badge model is supposed to be followed? That hidden layer of infrastructure must also be built securely, expand securely, and be maintained securely, otherwise the source code for those open source products and the security operations functions are put at risk. The CII Badge Program points to the GitHub Security policy and Heroku Security policy, both of which are great high-level lists of *what to do* (but not *how to do it* using any specific Linux distribution that a group would use to build a distributed system). What if a group can't or doesn't want to use GitHub and Heroku, or wants to operate within a *private cloud* deployment?

Not-for-profit or volunteer organizations who produce open source software, or those providing affordable managed security services to local government based on open source tools, must pay attention to the findings of the DBIR and address the infrastructure security requirements, while doing so on very limited budgets.

Those tasked with setting up and administering their own infrastructure, integrating multiple open source tools in a scalable or distributed deployment are not only faced with figuring out how to do it securely, but must also deal with the choices made by other teams who produced the open source tools to be integrated (which often are mutually exclusive in terms of base operating system distribution, OS release version, prerequisite libraries, and programming languages used by the services to be integrated.)

To sum up, we can paraphrase an old joke attributed to Jamie Zawinski (Source of the famous "Now you have two problems" quote):

> *Some people, when confronted with the problems just described, think "I know, I'll use open source security tools!" Now they have two problems.*

The remainder of this document will discuss *The Value Proposition*, *Licensing Plan*, and *Commercialization Plan*.

CHAPTER 2

The Value Proposition

As mentioned in *Introduction*, many of the products and services available in today's enterprise cybersecurity market have too many zeros in their total price. Nobody likes paying taxes, so local government can't afford expensive products or managed security services. Volunteers developing open source software don't want to give up both their time *and* the contents of their savings accounts.

This section discusses the *value proposition* for the products of the DIMS project.

The Need

You can't have good system security without good system administration. Organizations need to have strong system administration skills in order to have a secure foundation for their operations. That 1/3 of attacks due to mistakes and misconfigurations identified in Verizon's DBIR reflects a painful reality. And 100% of those breaches occurred in companies who employ humans.

Seriously, all humans make mistakes, or miss things. Or they may not know better when trying to just figure out how to get their job done and blindly follow someone's lead, opening themselves and their organization up to a major security hole (as seen in Fig. 2.1 from Don't Pipe to your Shell).

Mistakes are easier to make in situations where it is difficult to see what is going on, or where someone is forced to deal with something new that they have never dealt with before and have little expertise. Paul Vixie has described the pain (in terms of operations cost and impact on security posture) that results from *complexity* in today's distributed systems and security products. *[Vix16]*

> *Increased complexity without corresponding increases in understanding would be a net loss to a buyer. [...]*
>
> *The TCO of new technology products and services, including security-related products and services, should be fudge-factored by at least 3X to account for the cost of reduced understanding. That extra 2X is a source of new spending: on training, on auditing, on staff growth and retention, on in-house integration.*

As knowledge and experience increase, the quality of work output increases and the errors and omissions decrease. Finding and procuring the talent necessary to operate at the highest level, however, is neither easy, fast, nor cheap.

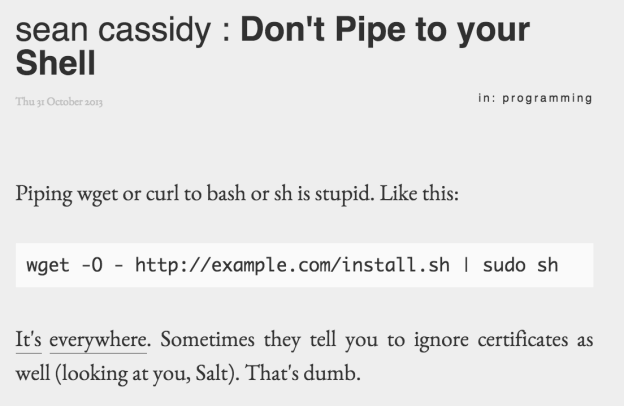

Fig. 2.1: Piping insecure content directly into a privileged shell

This all begs the question, "What can our organization do bring the capabilities of multiple open source products into a functioning whole with the least amount of pain and best operating security outcome?"

Our Approach

Our approach is to provide a reference model for establishing a secure and maintainable distributed open source platform that enables secure software development and secure system operations. The DIMS team (now implementing the third iteration of some of the core elements) has experienced the pain of this process, which will reduce the cost for those who adopt our methodology.

The DIMS project brings together multiple free/libre open source software (FOSS) tools in a reference model designed to be built securely from the ground up. The two primary outcomes of this effort are:

1. An example platform for building a complex integrated open source system for computer security incident response released as open source software and documentation. These products provide a working and documented model platform (or DevOps infrastructure) that can facilitate the secure integration of open source components that (in and of themselves) are often hard to deploy, and often are so insecurely implemented that they are effectively wide open to the internet. This not only solves some of the infrastructure problems alluded to by the Linux Foundation, but also addressing Vixie's example of supporting organizations wanting to use open source security tools in concert to address their trusted information sharing and security operations needs.
2. Transitioning this platform into the public sector to support operational needs of State, Local, Territorial, and Tribal (SLTT) government entities. DIMS project outputs are being evaluated by the PISCES-NW not-for-profit organization for use in the Pacific Northwest (see Section *PISCES Northwest*). The latest modification to the contract includes a pilot deployment for use by the United States Secret Service for their Electronic Crimes Task Force (ECTF) membership.

The dimssr:dimssystemrequirements documents security practices and features that we have incorporated to the greatest extent possible, in a way that can be improved over time in a modular manner. The system automation and continuous integration/continuous deployment (CI/CD) features help in implementing and maintaining a secure system. (Red team application penetration testing will further improve the security of the system through feedback about weaknesses and deficiencies that crept in during development and deployment.)

Golden nugget

Over two decades of system administration and security operations experience underlies the architectural model that we have been researching, developing, implementing, and documenting. The barrier to entry is the amount of time

and learning necessary to acquire this same expertise in order to be competitive.

Benefits per Cost

The value of the DIMS products and methodology comes from altering the cost equation described by Vixie, which can be expressed this way:

$$CustomerValue = \frac{CustomerBenefit}{cost(OpenSource) + cost(Implementation)}$$

The benefit to customers is maximized by the ability to construct and operate a secure incident response monitoring platform, expand it with additional open source tools as needed, saving a large part of the *2x* multiplier in implementation cost in system administration and operations overhead cited by Vixie. We enable this by helping make a less complex, more transparent, source controlled, and easier to secure open source platform than may otherwise be produced by someone leveraging multiple unfamiliar open source security tools from scratch. That means standing up a new server and adding new services to it can be reduced from taking hours or days per system to just a few minutes of effort. If that task has to be repeated dozens (or possibly hundreds) of times, the cost savings can be significant.

The DIMS team created and used a CI/CD model using Git, Jenkins CI, and Ansible for taking software source code, system automation instructions, software configuration, and documentation, to build a prototype for an open source software integration project. The resulting product can be used by an internal security operations group (or managed security service provider) to create an open source incident response capability. It also provides many of the elements called for in the CII Badge Program from the GitHub Security and Heroku Security policies.

Note: To see more detail about the full set of tools, techniques, and tasks that DIMS team members were expected to know or learn, see dimsjds:dimsjobdescriptions.

The impact of the effort expended in this project goes beyond implementing one set of open source service components for a single group. This model can be replicated widely and improved upon by others faced with the same set of challenges in developing an affordable and scalable incident response capability.

Note: Over the course of the project, we have learned of several other efforts to address a similar set of goals and have reached out (as time permitted) to find common ground and try to develop collaborative relationships that will have broad impact over time. This is expanded upon in Section *Commercialization Plan*.

Competition and Alternatives

The common way that organizations go about implementing open source products is by following whatever installation instructions may be provided by the authors. Avoiding the security problems illustrated by Fig. 2.1 involves searching the Internet to (hopefully) find some thread like Alternatives to piping the install script into your shell. #90 (from GitHub fisherman/fisherman, a "plugin manager for Fish," and no, we haven't heard of it before either.)

When it comes to the more difficult task of integrating multiple open source products into a functional distributed system, the research required to debug and solve an seemingly endless series of installation, configuration, and tuning problems.

Open Source Security Toolsets

Some of the open source security tools that an incident response team would want to consider implementing are covered in the following subsections.

Each of these systems is composed from several existing open source tools, combined with new open source scaffolding, glue, custom interfaces, and additional missing functionality that is necessary to achieve the resulting distributed system.

At the same time, each of these distributed open source systems relies upon their own chosen base operating system, libraries and languages, subordinate services (e.g., database, email transport agent, message bus, job scheduling, etc.) All too frequently, the choices made by each group are mutually exclusive, or left to the customer to work out on their own.

Note: To underscore Vixie's complexity and cost of implementation observation, Ubuntu 14.04 and Debian 7 have differences in how common services are configured that require debugging and custom configuration steps that vary between distributions, while the use of `systemd` for managing service daemons in Ubuntu 16.04 and Debian 8 are major impediments to migrating installation of all required components of these multi-service systems from Ubuntu 14.04 and Debian 7. Adding in RedHat Enterprise Linux, CentOS, or Fedora (all part of the same RedHat family) adds further complexity to the equation, which is a major reason why containerization is gaining popularity as a mechanism for isolating these dependency differences in a more manageable (but arguably less secure) fashion.

The Trident portal

The Trident portal is written in Go. Only Debian 7 (wheezy) is supported at this time, though Ubuntu 14.04 is on the list of future operating systems. Trident relies on PostreSQL for database, NGINX for web front end, and Postfix for email transport.

The Collective Intelligence Framework (CIF)

The Collective Intelligence Framework (CIF) is the primary offering from the CSIRT Gadgets Foundation. CIF is only supported on Ubuntu Linux. It is written in Perl and uses PostgreSQL, Apache2, BIND, Elasticsearch, ZeroMQ, and can support Kibana as an alternative interface to the indexed data in Elasticsearch.

A monolithic *EasyButton* installation script is available in the PlatformUbuntu section of the CIF wiki to automate the installation steps.

The Mozilla Defense Platform (MozDef)

The Mozilla Defense Platform (MozDef) was developed by Mozilla to replace a commercial SIEM product with open source alternatives. They report processing over 300 Million records per day with their internal deployment.

MozDef uses Ubuntu 14.04 as the base operating system. It has components for front-end user interface written in Javascript using Meteor, Node.js, and d3, and back-end data processing scripts written in Python using uWSGI, bottle.py, with MongoDB for a database, RabbitMQ for message bus, and NGINX for web app front end.

For installation, there is a demonstration `Dockerfile` for creating a monolithic Docker image with all of the MozDef components in it. (This is not the way Docker containers are intended to implement scalable microservices, but it does provide a very easy way to see a demonstration instance of MozDef). The manual instructions are more elaborate and must be followed carefully (including considering the admonitions related to security, e.g., "Configure your security group to open the ports you need. Keep in mind that it's probably a bad idea to have a public facing elasticsearch.")

GRR Rapid Response

Another example of a system made up of multiple components, packaged together into a single easy-to-install package, is the GRR Rapid Response system, a "forensic framework focused on scalability enabling powerful analysis."

GRR runs on Ubuntu 16.04. To ease installation of the server components, the GRR team, like CIF and MozDef, provide both a monolithic installation script for a VM installation and a `Dockerfile` to run in a container. They also have packages for installing the client components on Windows, OS X, and Linux.

> **Attention:** The GRR team chose to move to `systemd`, rather than continue to support the older `upstart`, `init.d`, or `supervisord` service daemon systems that are used by other products described in this section. This means you must support three (or four) different service daemon management mechanisms in order to incorporate all of the tools described here into a single integrated deployment.

GRR's documentation similarly includes admonitions about security and functionality that is left to the customer to implement. Take Fig. 2.2, a question from their FAQ as an example:

Fig. 2.2: Question about the logout button from GRR FAQ

Integrated Open Source Solutions

The DIMS project began in Q4 2013. In the second half of 2015 two very similar efforts were identified that use some of the same tools for the same reasons. Both validate the model being established by DIMS and the value proposition for adopters.

Summit Route Iterative Defense Architecture

An organization named Summit Route has described what they call the Iterative Defense Architecture (see Fig. 2.3) that is very similar in form and content to what the DIMS project has focused on producing.

OpenCredo

A consultancy in the United Kingdom named OpenCredo is also working on a similar architecture to the DIMS project (see Fig. 2.4). Some of the specific components differ, but conceptually are the same and would meet the same requirements for the foundation (minus the dashboard, portal, etc.) that is specified in dimssr:dimssystemrequirements.

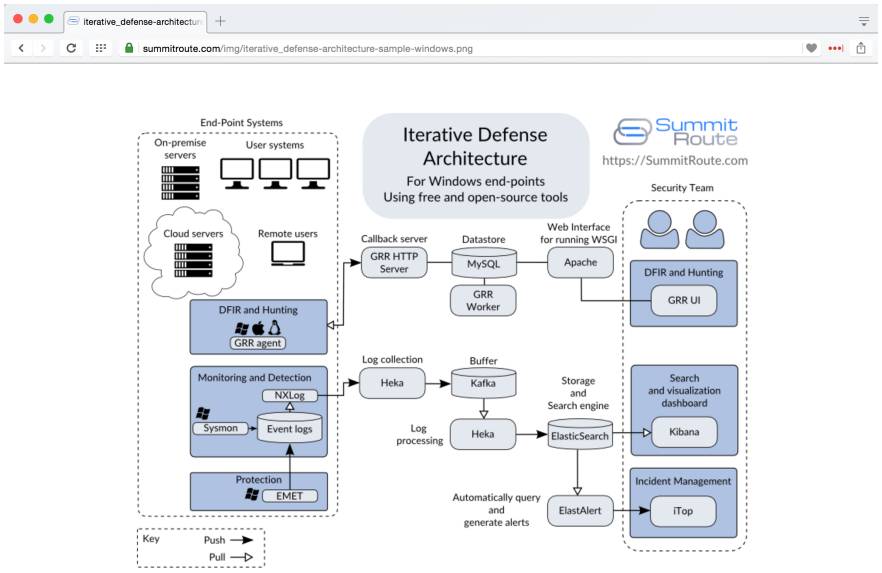

Fig. 2.3: Summit Route Integrated Defense Architecture

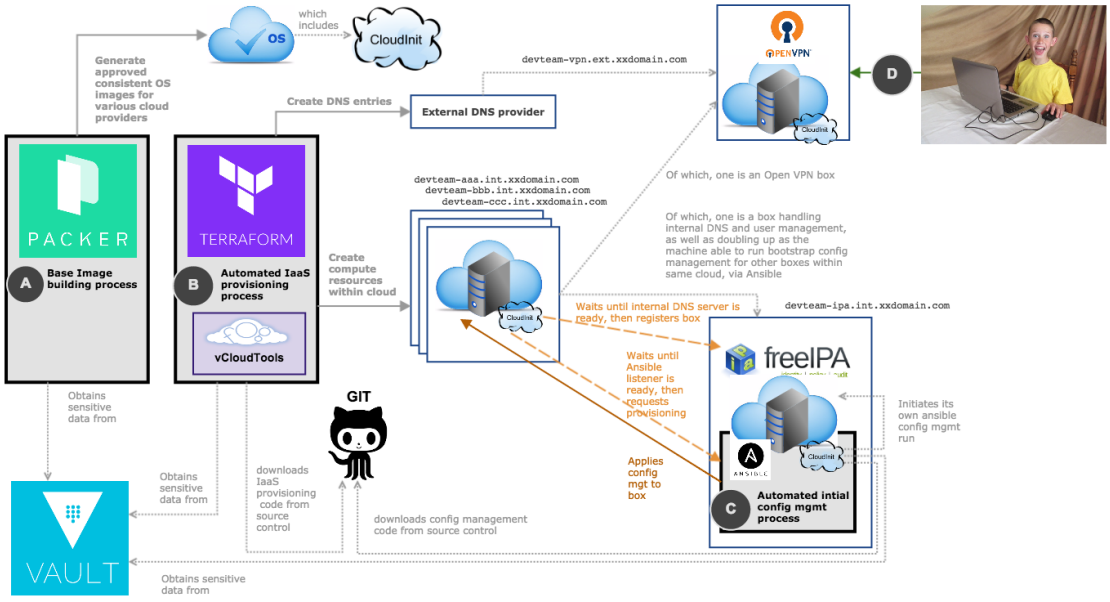

Fig. 2.4: OpenCredo core building blocks

CHAPTER 3

Licensing Plan

This section represents guidance obtained from an initial conversation in the Base Year with Peggy Hartman, and multiple conversations with Fred Holt in 2015 and 2016.

Compatibility of Open Source Licenses

Wikipedia describes a Permissive free software licence (sic), which is a concern when integrating open source projects. Figure Fig. 3.1 from this page is included here (under the terms of Creative Commons "Attribution-Share Alike 3.0 License").

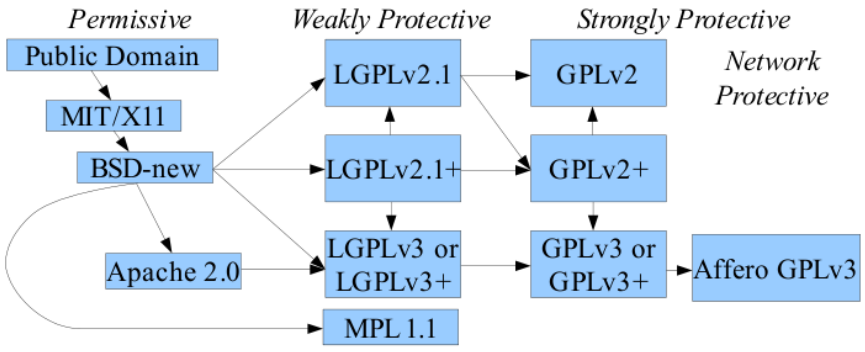

Fig. 3.1: License compatibility between common FOSS software licenses according to David A. Wheeler (2007)

University approved release license

Initial guidance from Peggy Hartman indicated that University of Washington preferred that software produced by the DIMS team be released under the Berkeley Three-Part license (also known as *BSD-3*). This is a simple license that falls into the *permissive* category of open source licenses.

For a copy of the license, see Section *License*.

Considerations for release of DIMS project source code

Attention: This subsection includes information obtained in conversations with Fred Holt, who has worked on intellectual property and technology transfer issues with the University of Washington Office of Technology Transfer.

The principal issue that Fred Holt stressed in these conversations has to do with *compatibility* of licenses, and the use of open source software that was released under a *restrictive* license within a larger project whose source is released under a *permissive* license.

Software licenses work in two directions: Source code is released by the copyright holder, along with restrictions and permissions on how it is to be used (or re-used); the recipient of open source software, wishing to re-use it and build a new or derived work that they intend to release must also choose a license under which their work is released, but must also adhere to the restrictions and permissions of the work they are basing their work upon. This raises issues of *compatibility* of terms in both licenses that an open source development team needs to be aware of and respect.

To understand the issues surrounding compatibility (or more importantly from a legal perspective, **incompatibility**) of open source licenses, a little history is necessary. The GPL has over 20 years of history of controversy for its terms and conditions. Some of this controversy resulted in a slightly less restrictive and narrower license known as the Lesser GPL (LGPL).

Part of the controversy over the GPL surrounds the time in which it came to exist and the style of programming languages at the time that were heavily slanted towards *compilation* and *linkage* (that is, writing *source code*, running a *compiler* on the source code to produce *object files* and *object libraries*, running a linker on the *object files and libraries* to produce an *executable binary image* (also commonly called an *executable* or *EXE* file). The GPL was targeted at these compiled executables, so if you used a `Makefile` (via the `make` command) to compile and link source code with libraries released under the GPL, then under the spirit of the GPL's terms, your program and its modules also had to be released under the GPL.

In today's programming environment, things are much more complicated. Languages like Python act more like interpreters than compilers, though they do produce a post-parsing form of binary code that helps speed execution by reducing redundant parsing. Modules are imported into Python programs (often by loading them into the directory hierarchy in which the Python interpreter stores its own module source files). Languages like Java similarly produce a *bytecode* intermediary binary format file that is executed by a bytecode interpreter known as the Java Virtual Machine (JVM), rather than being linked into a stand-alone executable, with modules used by the program.

This is complicated further as a result of the Unix philosophy of programs being simple and doing one thing very well, and those program being used with pipelining and other execution invocation mechanisms to compose these simple functions into higher-level more complex functionality, which can then be further combined, and on and on. This raises questions like, "Is a program that forks a GPL-licensed program via a system or vice versa call derivative work?" and – specifically to the DIMS project, which uses Ansible – "[Does] the GPL license imply that my [Ansible] plugins are also GPL?" (see Ansible issue #8864) According to Holt and these commenters, simply using GPL code does not always trigger the "derivative work" clause, but care must still be taken to be clear about respect for the intent of these restrictive licenses.

Abiding the Spirit of Restrictive Licenses

Given the spirit of intent of the GPL, the following guidance will be applied to DIMS code:

- If source code is compiled and a GPL licensed module linked into it, then the resulting executable should be released under the GPL.

Note: The exception is the case where the GPL code may just facilitate one "layer" of a complex program, where the LGPL is more applicable.

- If we invoke a GPL program in a shell (even with arguments) then the other levels of the larger program, above and below the shell that invokes the GPL program, can be released under a different license (e.g. BSD-3)

Attention: While developing and integrating open source products, it is important to not only be aware of code released under GPL, but also look for a good boundary around GPL licensed code that respects the spirit of the license.

Implementing separation in source code

Fred Holt described two options (*Gold* and *Purple*, two arbitrarily chosen colors that have *nothing* to do with the University of Washington's school colors) for handling licensing text and notification in source code repositories.

Fig. 3.2 (*Option Gold*) shows three different source directories:

- Left side: BSD Three-Part licensed code using a common header `hdr_bsd`
- Middle: "BSD (from Apache with notice)" using a common header `hdr_bsd_from_apache`
- Right side: Apache 2 licensed source code using a common header `hdr_apache`

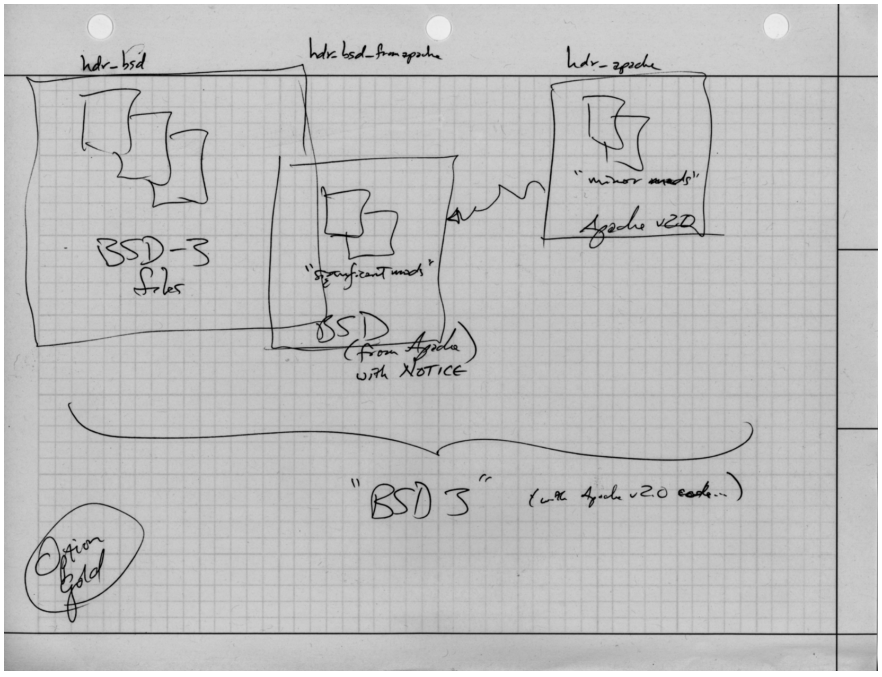

Fig. 3.2: 'Option Gold' strategy for handling source code and selecting release license

This option is for dealing with primarily new BSD-3 code that relies in part on Apache 2 source code that has been modified to some degree. The degree to which is has been modified, while not a clear black-and-white determination, informs whether the derived source should be released under the original license of the source work (i.e., Apache 2), or under the desired Berkeley Three-Part license.

- The middle section is for *significant mods* to the original code. An example of a minor modification would be renaming variables to match naming conventions used in the DIMS project and changing values to brand the resulting run-time interfaces to match DIMS branding.
- The right section is for *minor mods* to the code. A major modification would be fundamentally altering functions, classes, or adding substantial new code.

Either way, the original code is clearly identified as being distinct from newly written code, the author of the code is acknowledged, and a notice is included that the major work is derived from existing code and the license under which that code was originally released.

Fig. 3.3 (*Option Purple*) shows how Apache 2 licensed code (kept separate on the right side of the figure) is included unmodified in a separate directory that isolates it from DIMS code with BSD (kept separate on the left). The *new works* code, for example sub-classes that inherit from parent classes in the Apache 2 code base, are shown within the BSD box. The repository is then released with the original license from the Apache 2 code base and a note that the major work is BSD-3 that uses Apache 2 components (along with references to where the code was originally obtained.)

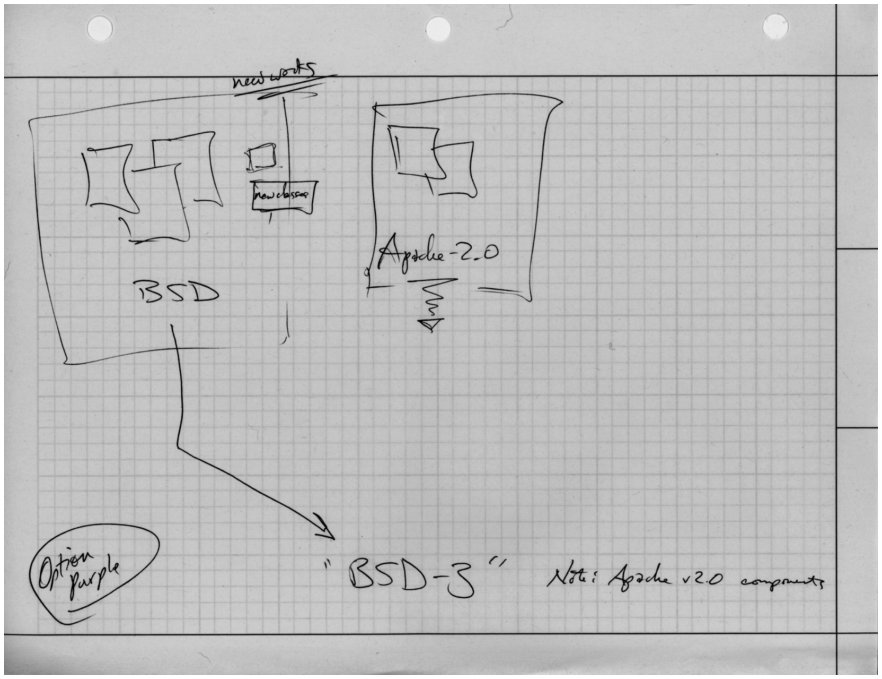

Fig. 3.3: 'Option Purple' strategy for handling source code and selecting release license

CHAPTER 4

Commercialization Plan

This section describes some options for commercialization and technology transfer that are being considered for DIMS deliverable products.

Intellectual Property

Under the terms of the Contract, Section C.3.4, the products of this project are to be released as open source under a selected open source license. As described in Section *University approved release license*, this license is the Berkeley Three-Part license.

Technology Transfer

No intellectual property disclosures to the University of Washington will result from this project. All technology transfer will result from uniform public access to the released open source code and documentation. That said, getting the open source products resulting from this contract to be widely used will not be simple, or easy. As Maughan *et al [MBLT13]* discuss, projects do not sell themselves and many attempts may be necessary (some resulting in failure) before success is achieved.

Outreach activities, and collaboration attempts during the project to date have shown that language, pictures, shared experience, and a clear description of the problems and proposed solution are important (yet simultaneously a challenge to achieve.) Still, the conversations we have had with multiple organizations are promising. Following sections list some of the organizations that have been approached about using, continuing to develop, or promoting the open source products resulting from this contract.

CSIRT Gadgets Foundation

Conversations with Wes Young and Gabe Iovino of the CSIRT Gadgets Foundation indicate that their foundation may be a good place for forks of the DIMS code, configuration, and documentation repositories to be housed and maintained similarly to the way the Collective Intelligence Framework is maintained. Additionally, there are opportunities

working with the foundation to enhance CIF using DIMS products and lessons learned. This would be a natural place to take the techniques in system administration automation, Docker containerization and CoreOS clustering, and continuous integration of source components and system configuration.

Farsight Security

Farsight Security has expressed an interest in supporting continued development of DIMS components with letters of support and other political and social acts, but desires to be a client in future collaborations rather than a volunteer contributor. Further conversations with Farsight may explore possible interest in grants or contracts to provide financial support for further system integration efforts.

Farsight has been very generous in making architectural changes to the new (and soon-to-be publicly released) **Trident** portal system that enable DIMS component integration with Trident. The DIMS team has been working with Farsight to facilitate red team application assessment that will help improve Trident.

PISCES Northwest

A not-for-profit entity known as the Public Infrastructure Security Collaboration and Exchange System (PISCES-NW, for "North West") was recently formed. The Board of Directors is seeking grant funding to extend a regional SLTT security monitoring project (formerly known as the Public Regional Information Security Event Monitoring project, or "PRISEM"). Under this proposal, the PI will be engaged for a limited time in the initial phase as a sub-contractor, focused on assisting with implementation of selected DIMS open source products as requested by PISCES-NW. One of the PISCES-NW project's objectives is to integrate DHS S&T-funded research products, which fits in line with values and objectives described in Sections *Introduction* and *The Value Proposition*.

Cyber Resilience Institute

The PI was invited to be on the Board of Directors of the Colorado-based Cyber Resilience Institute (CRI). DIMS products will be demonstrated to the CRI Board and considered for inclusion in pilot projects that CRI is pursuing, possibly in collaboration with educational institutions in the state of Colorado.

Other Security Companies

Conversations have taken place with other "stealth-mode" computer security companies, both in Washington state and elsewhere. Because of non-disclosure agreements, they will not be directly named here. The discussions have involved the possibility of using and contributing back to the DIMS open source code products, using them to complement internally-developed commercial products and services, and/or implementing custom deployments of DIMS+Trident components for customers to use in forming and operating trusted information sharing and security operations. Possible partnership between several of these companies is on the table, which could greatly accelerate continued development of products resulting from the DIMS contract.

CHAPTER 5

License

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

Copyright (c) 2014, 2016 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.
```

CHAPTER 6

Contact:

Section author: David Dittrich dittrich@u.washington.edu

Bibliography

[DBIR16] Verizon. 2016 Data Breach Investigations Report. http://www.verizonenterprise.com/verizon-insights-lab/dbir/2016/, April 2016.

[Mor15] Steve Morgan. Cybersecurity Market Reaches $75 Billion In 2015; Expected To Reach $170 Billion By 2020. http://onforb.es/1QDaK3D, December 2015.

[Vix16] Paul Vixie. Magical Thinking in Internet Security. https://www.farsightsecurity.com/Blog/20160428-vixie-magicalthinking/, April 2016.

[MBLT13] Douglas Maughan, David Balenson, Ulf Lindqvist, and Zachary Tudor. Crossing the "Valley of Death": Transitioning Cybersecurity Research into Practice. IEEE Security & Privacy, 11(2):14–23, 2013.