



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

| APPLICATION<br>NUMBER | FILING or<br>371(c) DATE | GRP ART<br>UNIT | FIL FEE REC'D | ATTY. DOCKET NO. | TOT CLAIMS | IND CLAIMS |
|-----------------------|--------------------------|-----------------|---------------|------------------|------------|------------|
| 62/590,986            | 11/27/2017               |                 | 260           | LACEP001+        |            |            |

**CONFIRMATION NO. 1075**  
**FILING RECEIPT**



CC000000093912840

21912  
VAN PELT, YI & JAMES LLP  
10050 N. FOOTHILL BLVD #200  
CUPERTINO, CA 95014

Date Mailed: 12/07/2017

Receipt is acknowledged of this provisional patent application. It will not be examined for patentability and will become abandoned not later than twelve months after its filing date. Any correspondence concerning the application must include the following identification information: the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please submit a written request for a Filing Receipt Correction. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections**

**Inventor(s)**

Vikram Kapoor, Cupertino, CA;  
Rakesh Sachdeva, Santa Clara, CA;  
Samuel Joseph Pullara III, Los Altos, CA;  
Murat Bog, Fremont, CA;  
Yijou Chen, Cupertino, CA;  
Sanjay Kalra, San Jose, CA;  
Harish Kumar Bharat Singh, Mountain View, CA;

**Applicant(s)**

Lacework Inc., Mountain View, CA

**Power of Attorney:**

Robyn Wagner--50575

**Permission to Access Application via Priority Document Exchange:** Yes

**Permission to Access Search Results:** Yes

Applicant may provide or rescind an authorization for access using Form PTO/SB/39 or Form PTO/SB/69 as appropriate.

**If Required, Foreign Filing License Granted:** 12/06/2017

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US 62/590,986**

**Projected Publication Date:** None, application is not eligible for pre-grant publication

page 1 of 3

**Non-Publication Request:** No

**Early Publication Request:** No

**Title**

USING GRAPH-BASED MODELS TO IDENTIFY ANOMALIES

**Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications:** No

## **PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES**

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process **simplifies** the filing of patent applications on the same invention in member countries, but **does not result** in a grant of "an international patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at <http://www.uspto.gov/web/offices/pac/doc/general/index.html>.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, <http://www.stopfakes.gov>. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4258).

## **LICENSE FOR FOREIGN FILING UNDER**

**Title 35, United States Code, Section 184**

**Title 37, Code of Federal Regulations, 5.11 & 5.15**

### **GRANTED**

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where

page 2 of 3

the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign AssetsControl, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

#### **NOT GRANTED**

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

---

### **SelectUSA**

The United States represents the largest, most dynamic marketplace in the world and is an unparalleled location for business investment, innovation, and commercialization of new technologies. The U.S. offers tremendous resources and advantages for those who invest and manufacture goods here. Through SelectUSA, our nation works to promote and facilitate business investment. SelectUSA provides information assistance to the international investor community; serves as an ombudsman for existing and potential investors; advocates on behalf of U.S. cities, states, and regions competing for global investment; and counsels U.S. economic development organizations on investment attraction best practices. To learn more about why the United States is the best country in the world to develop technology, manufacture products, deliver services, and grow your business, visit <http://www.SelectUSA.gov> or call +1-202-482-6800.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                                                |                        |           |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------|------------------------|-----------|
| <b>Application Data Sheet 37 CFR 1.76</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                                                | Attorney Docket Number | LACEP001+ |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                                                | Application Number     |           |
| Title of Invention                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | USING GRAPH-BASED MODELS TO IDENTIFY ANOMALIES |                        |           |
| <p>The application data sheet is part of the provisional or nonprovisional application for which it is being submitted. The following form contains the bibliographic data arranged in a format specified by the United States Patent and Trademark Office as outlined in 37 CFR 1.76.</p> <p>This document may be completed electronically and submitted to the Office in electronic format using the Electronic Filing System (EFS) or the document may be printed and included in a paper filed application.</p> |                                                |                        |           |

**Secrecy Order 37 CFR 5.2:**

☐ Portions or all of the application associated with this Application Data Sheet may fall under a Secrecy Order pursuant to 37 CFR 5.2 (Paper filers only. Applications that fall under Secrecy Order may not be filed electronically.)

**Inventor Information:**

|                                                                                                                                                                          |                      |                |             |                      |        |  |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|----------------|-------------|----------------------|--------|--|
| Inventor                                                                                                                                                                 | 1                    |                |             |                      | Remove |  |
| Legal Name                                                                                                                                                               |                      |                |             |                      |        |  |
| Prefix                                                                                                                                                                   | Given Name           | Middle Name    | Family Name | Suffix               |        |  |
|                                                                                                                                                                          | Vikram               |                | Kapoor      |                      |        |  |
| Residence Information (Select One) <input checked="" type="radio"/> US Residency <input type="radio"/> Non US Residency <input type="radio"/> Active US Military Service |                      |                |             |                      |        |  |
| City                                                                                                                                                                     | Cupertino            | State/Province | CA          | Country of Residence | US     |  |
| Mailing Address of Inventor:                                                                                                                                             |                      |                |             |                      |        |  |
| Address 1                                                                                                                                                                | 700 E El Camino Real |                |             |                      |        |  |
| Address 2                                                                                                                                                                | Suite 130            |                |             |                      |        |  |
| City                                                                                                                                                                     | Mountain View        | State/Province | CA          |                      |        |  |
| Postal Code                                                                                                                                                              | 94041                | Country i      | US          |                      |        |  |
| Inventor                                                                                                                                                                 | 2                    |                |             |                      | Remove |  |
| Legal Name                                                                                                                                                               |                      |                |             |                      |        |  |
| Prefix                                                                                                                                                                   | Given Name           | Middle Name    | Family Name | Suffix               |        |  |
|                                                                                                                                                                          | Rakesh               |                | Sachdeva    |                      |        |  |
| Residence Information (Select One) <input checked="" type="radio"/> US Residency <input type="radio"/> Non US Residency <input type="radio"/> Active US Military Service |                      |                |             |                      |        |  |
| City                                                                                                                                                                     | Santa Clara          | State/Province | CA          | Country of Residence | US     |  |
| Mailing Address of Inventor:                                                                                                                                             |                      |                |             |                      |        |  |
| Address 1                                                                                                                                                                | 700 E El Camino Real |                |             |                      |        |  |
| Address 2                                                                                                                                                                | Suite 130            |                |             |                      |        |  |
| City                                                                                                                                                                     | Mountain View        | State/Province | CA          |                      |        |  |
| Postal Code                                                                                                                                                              | 94041                | Country i      | US          |                      |        |  |
| Inventor                                                                                                                                                                 | 3                    |                |             |                      | Remove |  |
| Legal Name                                                                                                                                                               |                      |                |             |                      |        |  |
| Prefix                                                                                                                                                                   | Given Name           | Middle Name    | Family Name | Suffix               |        |  |
|                                                                                                                                                                          | Samuel               | Joseph         | Pullara     | III                  |        |  |
| Residence Information (Select One) <input checked="" type="radio"/> US Residency <input type="radio"/> Non US Residency <input type="radio"/> Active US Military Service |                      |                |             |                      |        |  |

|                                           |                                                |                        |           |
|-------------------------------------------|------------------------------------------------|------------------------|-----------|
| <b>Application Data Sheet 37 CFR 1.76</b> |                                                | Attorney Docket Number | LACEP001+ |
|                                           |                                                | Application Number     |           |
| Title of Invention                        | USING GRAPH-BASED MODELS TO IDENTIFY ANOMALIES |                        |           |

|      |           |                |    |                      |    |
|------|-----------|----------------|----|----------------------|----|
| City | Los Altos | State/Province | CA | Country of Residence | US |
|------|-----------|----------------|----|----------------------|----|

**Mailing Address of Inventor:**

|             |                      |                |    |  |  |
|-------------|----------------------|----------------|----|--|--|
| Address 1   | 700 E El Camino Real |                |    |  |  |
| Address 2   | Suite 130            |                |    |  |  |
| City        | Mountain View        | State/Province | CA |  |  |
| Postal Code | 94041                | Country        | US |  |  |

|          |   |                        |
|----------|---|------------------------|
| Inventor | 4 | <a href="#">Remove</a> |
|----------|---|------------------------|

**Legal Name**

| Prefix | Given Name | Middle Name | Family Name | Suffix |
|--------|------------|-------------|-------------|--------|
|        | Murat      |             | Bog         |        |

|                                           |                                               |                                        |                                                  |
|-------------------------------------------|-----------------------------------------------|----------------------------------------|--------------------------------------------------|
| <b>Residence Information (Select One)</b> | <input checked="" type="radio"/> US Residency | <input type="radio"/> Non US Residency | <input type="radio"/> Active US Military Service |
|-------------------------------------------|-----------------------------------------------|----------------------------------------|--------------------------------------------------|

|      |         |                |    |                      |    |
|------|---------|----------------|----|----------------------|----|
| City | Fremont | State/Province | CA | Country of Residence | US |
|------|---------|----------------|----|----------------------|----|

**Mailing Address of Inventor:**

|             |                      |                |    |  |  |
|-------------|----------------------|----------------|----|--|--|
| Address 1   | 700 E El Camino Real |                |    |  |  |
| Address 2   | Suite 130            |                |    |  |  |
| City        | Mountain View        | State/Province | CA |  |  |
| Postal Code | 94041                | Country        | US |  |  |

|          |   |                        |
|----------|---|------------------------|
| Inventor | 5 | <a href="#">Remove</a> |
|----------|---|------------------------|

**Legal Name**

| Prefix | Given Name | Middle Name | Family Name | Suffix |
|--------|------------|-------------|-------------|--------|
|        | Yijou      |             | Chen        |        |

|                                           |                                               |                                        |                                                  |
|-------------------------------------------|-----------------------------------------------|----------------------------------------|--------------------------------------------------|
| <b>Residence Information (Select One)</b> | <input checked="" type="radio"/> US Residency | <input type="radio"/> Non US Residency | <input type="radio"/> Active US Military Service |
|-------------------------------------------|-----------------------------------------------|----------------------------------------|--------------------------------------------------|

|      |           |                |    |                      |    |
|------|-----------|----------------|----|----------------------|----|
| City | Cupertino | State/Province | CA | Country of Residence | US |
|------|-----------|----------------|----|----------------------|----|

**Mailing Address of Inventor:**

|             |                      |                |    |  |  |
|-------------|----------------------|----------------|----|--|--|
| Address 1   | 700 E El Camino Real |                |    |  |  |
| Address 2   | Suite 130            |                |    |  |  |
| City        | Mountain View        | State/Province | CA |  |  |
| Postal Code | 94041                | Country        | US |  |  |

|          |   |                        |
|----------|---|------------------------|
| Inventor | 6 | <a href="#">Remove</a> |
|----------|---|------------------------|

**Legal Name**

| Prefix | Given Name | Middle Name | Family Name | Suffix |
|--------|------------|-------------|-------------|--------|
|        | Sanjay     |             | Kalra       |        |

|                                           |                                               |                                        |                                                  |
|-------------------------------------------|-----------------------------------------------|----------------------------------------|--------------------------------------------------|
| <b>Residence Information (Select One)</b> | <input checked="" type="radio"/> US Residency | <input type="radio"/> Non US Residency | <input type="radio"/> Active US Military Service |
|-------------------------------------------|-----------------------------------------------|----------------------------------------|--------------------------------------------------|

|      |          |                |    |                      |    |
|------|----------|----------------|----|----------------------|----|
| City | San Jose | State/Province | CA | Country of Residence | US |
|------|----------|----------------|----|----------------------|----|

|                                           |                                                |                        |           |
|-------------------------------------------|------------------------------------------------|------------------------|-----------|
| <b>Application Data Sheet 37 CFR 1.76</b> |                                                | Attorney Docket Number | LACEP001+ |
|                                           |                                                | Application Number     |           |
| Title of Invention                        | USING GRAPH-BASED MODELS TO IDENTIFY ANOMALIES |                        |           |

|                                                                                                                                                                                 |                      |                |             |                                       |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|----------------|-------------|---------------------------------------|
| <b>Mailing Address of Inventor:</b>                                                                                                                                             |                      |                |             |                                       |
| Address 1                                                                                                                                                                       | 700 E El Camino Real |                |             |                                       |
| Address 2                                                                                                                                                                       | Suite 130            |                |             |                                       |
| City                                                                                                                                                                            | Mountain View        | State/Province | CA          |                                       |
| Postal Code                                                                                                                                                                     | 94041                | Country i      | US          |                                       |
| Inventor                                                                                                                                                                        | 7                    |                |             | <input type="button" value="Remove"/> |
| <b>Legal Name</b>                                                                                                                                                               |                      |                |             |                                       |
| Prefix                                                                                                                                                                          | Given Name           | Middle Name    | Family Name | Suffix                                |
|                                                                                                                                                                                 | Harish               | Kumar Bharat   | Singh       |                                       |
| <b>Residence Information (Select One)</b> <input checked="" type="radio"/> US Residency <input type="radio"/> Non US Residency <input type="radio"/> Active US Military Service |                      |                |             |                                       |
| City                                                                                                                                                                            | Mountain View        | State/Province | CA          | Country of Residence                  |
|                                                                                                                                                                                 |                      | US             |             |                                       |

|                                                                                                                                                                                |                      |                |    |  |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|----------------|----|--|
| <b>Mailing Address of Inventor:</b>                                                                                                                                            |                      |                |    |  |
| Address 1                                                                                                                                                                      | 700 E El Camino Real |                |    |  |
| Address 2                                                                                                                                                                      | Suite 130            |                |    |  |
| City                                                                                                                                                                           | Mountain View        | State/Province | CA |  |
| Postal Code                                                                                                                                                                    | 94041                | Country i      | US |  |
| All Inventors Must Be Listed - Additional Inventor Information blocks may be generated within this form by selecting the <b>Add</b> button. <input type="button" value="Add"/> |                      |                |    |  |

### Correspondence Information:

|                                                                                                                                       |                                                                                      |
|---------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| Enter either Customer Number or complete the Correspondence Information section below.<br>For further information see 37 CFR 1.33(a). |                                                                                      |
| <input type="checkbox"/> An Address is being provided for the correspondence Information of this application.                         |                                                                                      |
| Customer Number                                                                                                                       | 21912                                                                                |
| Email Address                                                                                                                         | <input type="button" value="Add Email"/> <input type="button" value="Remove Email"/> |

### Application Information:

|                                         |                                                |                                           |                          |
|-----------------------------------------|------------------------------------------------|-------------------------------------------|--------------------------|
| Title of the Invention                  | USING GRAPH-BASED MODELS TO IDENTIFY ANOMALIES |                                           |                          |
| Attorney Docket Number                  | LACEP001+                                      | Small Entity Status Claimed               | <input type="checkbox"/> |
| Application Type                        | Provisional                                    |                                           |                          |
| Subject Matter                          | Utility                                        |                                           |                          |
| Total Number of Drawing Sheets (if any) | 19                                             | Suggested Figure for Publication (if any) |                          |

|                                           |                                                |                        |           |
|-------------------------------------------|------------------------------------------------|------------------------|-----------|
| <b>Application Data Sheet 37 CFR 1.76</b> |                                                | Attorney Docket Number | LACEP001+ |
|                                           |                                                | Application Number     |           |
| Title of Invention                        | USING GRAPH-BASED MODELS TO IDENTIFY ANOMALIES |                        |           |

### Filing By Reference:

Only complete this section when filing an application by reference under 35 U.S.C. 111(c) and 37 CFR 1.57(a). Do not complete this section if application papers including a specification and any drawings are being filed. Any domestic benefit or foreign priority information must be provided in the appropriate section(s) below (i.e., "Domestic Benefit/National Stage Information" and "Foreign Priority Information").

For the purposes of a filing date under 37 CFR 1.53(b), the description and any drawings of the present application are replaced by this reference to the previously filed application, subject to conditions and requirements of 37 CFR 1.57(a).

|                                                        |                          |                                            |
|--------------------------------------------------------|--------------------------|--------------------------------------------|
| Application number of the previously filed application | Filing date (YYYY-MM-DD) | Intellectual Property Authority or Country |
|                                                        |                          |                                            |

### Publication Information:

☐ Request Early Publication (Fee required at time of Request 37 CFR 1.219)

☒ **Request Not to Publish.** I hereby request that the attached application not be published under 35 U.S.C. 122(b) and certify that the invention disclosed in the attached application **has not and will not** be the subject of an application filed in another country, or under a multilateral international agreement, that requires publication at eighteen months after filing.

### Representative Information:

Representative information should be provided for all practitioners having a power of attorney in the application. Providing this information in the Application Data Sheet does not constitute a power of attorney in the application (see 37 CFR 1.32). Either enter Customer Number or complete the Representative Name section below. If both sections are completed the customer Number will be used for the Representative Information during processing.

|                    |                                                  |                                              |                                                         |
|--------------------|--------------------------------------------------|----------------------------------------------|---------------------------------------------------------|
| Please Select One: | <input checked="" type="radio"/> Customer Number | <input type="radio"/> US Patent Practitioner | <input type="radio"/> Limited Recognition (37 CFR 11.9) |
| Customer Number    | 21912                                            |                                              |                                                         |

### Domestic Benefit/National Stage Information:

This section allows for the applicant to either claim benefit under 35 U.S.C. 119(e), 120, 121, 365(c), or 386(c) or indicate National Stage entry from a PCT application. Providing benefit claim information in the Application Data Sheet constitutes the specific reference required by 35 U.S.C. 119(e) or 120, and 37 CFR 1.78.

When referring to the current application, please leave the "Application Number" field blank.

|                                                                                                                           |                 |                          |                                    |
|---------------------------------------------------------------------------------------------------------------------------|-----------------|--------------------------|------------------------------------|
| Prior Application Status                                                                                                  | <div></div>     | <div>Remove</div>        |                                    |
| Application Number                                                                                                        | Continuity Type | Prior Application Number | Filing or 371(c) Date (YYYY-MM-DD) |
| <div></div>                                                                                                               | <div></div>     |                          |                                    |
| Additional Domestic Benefit/National Stage Data may be generated within this form by selecting the <div>Add</div> button. |                 |                          |                                    |

|                                           |                                                |                        |           |
|-------------------------------------------|------------------------------------------------|------------------------|-----------|
| <b>Application Data Sheet 37 CFR 1.76</b> |                                                | Attorney Docket Number | LACEP001+ |
|                                           |                                                | Application Number     |           |
| Title of Invention                        | USING GRAPH-BASED MODELS TO IDENTIFY ANOMALIES |                        |           |

### Foreign Priority Information:

This section allows for the applicant to claim priority to a foreign application. Providing this information in the application data sheet constitutes the claim for priority as required by 35 U.S.C. 119(b) and 37 CFR 1.55. When priority is claimed to a foreign application that is eligible for retrieval under the priority document exchange program (PDX)<sup>i</sup> the information will be used by the Office to automatically attempt retrieval pursuant to 37 CFR 1.55(i)(1) and (2). Under the PDX program, applicant bears the ultimate responsibility for ensuring that a copy of the foreign application is received by the Office from the participating foreign intellectual property office, or a certified copy of the foreign priority application is filed, within the time period specified in 37 CFR 1.55(g)(1).

|                                                                                                        |                      |                          | <a href="#">Remove</a>                   |
|--------------------------------------------------------------------------------------------------------|----------------------|--------------------------|------------------------------------------|
| Application Number                                                                                     | Country <sup>i</sup> | Filing Date (YYYY-MM-DD) | Access Code <sup>i</sup> (if applicable) |
|                                                                                                        |                      |                          |                                          |
| Additional Foreign Priority Data may be generated within this form by selecting the <b>Add</b> button. |                      |                          | <a href="#">Add</a>                      |

### Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>This application (1) claims priority to or the benefit of an application filed before March 16, 2013 and (2) also contains, or contained at any time, a claim to a claimed invention that has an effective filing date on or after March 16, 2013.</p> <p><input type="checkbox"/> NOTE: By providing this statement under 37 CFR 1.55 or 1.78, this application, with a filing date on or after March 16, 2013, will be examined under the first inventor to file provisions of the AIA.</p> |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|                                           |                                                |                        |           |
|-------------------------------------------|------------------------------------------------|------------------------|-----------|
| <b>Application Data Sheet 37 CFR 1.76</b> |                                                | Attorney Docket Number | LACEP001+ |
|                                           |                                                | Application Number     |           |
| Title of Invention                        | USING GRAPH-BASED MODELS TO IDENTIFY ANOMALIES |                        |           |

## Authorization or Opt-Out of Authorization to Permit Access:

When this Application Data Sheet is properly signed and filed with the application, applicant has provided written authority to permit a participating foreign intellectual property (IP) office access to the instant application-as-filed (see paragraph A in subsection 1 below) and the European Patent Office (EPO) access to any search results from the instant application (see paragraph B in subsection 1 below).

Should applicant choose not to provide an authorization identified in subsection 1 below, applicant **must opt-out** of the authorization by checking the corresponding box A or B or both in subsection 2 below.

**NOTE:** This section of the Application Data Sheet is **ONLY** reviewed and processed with the **INITIAL** filing of an application. After the initial filing of an application, an Application Data Sheet cannot be used to provide or rescind authorization for access by a foreign IP office(s). Instead, Form PTO/SB/39 or PTO/SB/69 must be used as appropriate.

### 1. Authorization to Permit Access by a Foreign Intellectual Property Office(s)

**A. Priority Document Exchange (PDX)** - Unless box A in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the European Patent Office (EPO), the Japan Patent Office (JPO), the Korean Intellectual Property Office (KIPO), the State Intellectual Property Office of the People's Republic of China (SIPO), the World Intellectual Property Organization (WIPO), and any other foreign intellectual property office participating with the USPTO in a bilateral or multilateral priority document exchange agreement in which a foreign application claiming priority to the instant patent application is filed, access to: (1) the instant patent application-as-filed and its related bibliographic data, (2) any foreign or domestic application to which priority or benefit is claimed by the instant application and its related bibliographic data, and (3) the date of filing of this Authorization. See 37 CFR 1.14(h)(1).

**B. Search Results from U.S. Application to EPO** - Unless box B in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the EPO access to the bibliographic data and search results from the instant patent application when a European patent application claiming priority to the instant patent application is filed. See 37 CFR 1.14(h)(2).

The applicant is reminded that the EPO's Rule 141(1) EPC (European Patent Convention) requires applicants to submit a copy of search results from the instant application without delay in a European patent application that claims priority to the instant application.

### 2. Opt-Out of Authorizations to Permit Access by a Foreign Intellectual Property Office(s)

☐ A. Applicant **DOES NOT** authorize the USPTO to permit a participating foreign IP office access to the instant application-as-filed. If this box is checked, the USPTO will not be providing a participating foreign IP office with any documents and information identified in subsection 1A above.

☐ B. Applicant **DOES NOT** authorize the USPTO to transmit to the EPO any search results from the instant patent application. If this box is checked, the USPTO will not be providing the EPO with search results from the instant application.

**NOTE:** Once the application has published or is otherwise publicly available, the USPTO may provide access to the application in accordance with 37 CFR 1.14.

|                                           |                                                |                        |           |
|-------------------------------------------|------------------------------------------------|------------------------|-----------|
| <b>Application Data Sheet 37 CFR 1.76</b> |                                                | Attorney Docket Number | LACEP001+ |
|                                           |                                                | Application Number     |           |
| Title of Invention                        | USING GRAPH-BASED MODELS TO IDENTIFY ANOMALIES |                        |           |

## Applicant Information:

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                          |                                                                        |                |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------|------------------------------------------------------------------------|----------------|
| Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                          |                                                                        |                |
| <b>Applicant</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 1 <span>Remove</span>                    |                                                                        |                |
| If the applicant is the inventor (or the remaining joint inventor or inventors under 37 CFR 1.45), this section should not be completed. The information to be provided in this section is the name and address of the legal representative who is the applicant under 37 CFR 1.43; or the name and address of the assignee, person to whom the inventor is under an obligation to assign the invention, or person who otherwise shows sufficient proprietary interest in the matter who is the applicant under 37 CFR 1.46. If the applicant is an applicant under 37 CFR 1.46 (assignee, person to whom the inventor is obligated to assign, or person who otherwise shows sufficient proprietary interest) together with one or more joint inventors, then the joint inventor or inventors who are also the applicant should be identified in this section. <span>Clear</span> |                                          |                                                                        |                |
| Assignee                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Legal Representative under 35 U.S.C. 117 |                                                                        | Joint Inventor |
| <input type="radio"/> Person to whom the inventor is obligated to assign.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                          | <input type="radio"/> Person who shows sufficient proprietary interest |                |
| If applicant is the legal representative, indicate the authority to file the patent application, the inventor is:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                          |                                                                        |                |
| <div></div>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                          |                                                                        |                |
| Name of the Deceased or Legally Incapacitated Inventor: <div></div>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                          |                                                                        |                |
| If the Applicant is an Organization check here. <input checked="" type="checkbox"/>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                          |                                                                        |                |
| Organization Name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Lacework Inc.                            |                                                                        |                |
| <b>Mailing Address Information For Applicant:</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                          |                                                                        |                |
| Address 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 700 E El Camino Real                     |                                                                        |                |
| Address 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Suite 130                                |                                                                        |                |
| City                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Mountain View                            | State/Province                                                         | CA             |
| Country                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | US                                       | Postal Code                                                            | 94041          |
| Phone Number                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                                          | Fax Number                                                             |                |
| Email Address                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                                          |                                                                        |                |
| Additional Applicant Data may be generated within this form by selecting the Add button. <span>Add</span>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                          |                                                                        |                |

## Assignee Information including Non-Applicant Assignee Information:

|                                                                                                                                                                                     |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office. |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|                                           |                                                |                        |           |
|-------------------------------------------|------------------------------------------------|------------------------|-----------|
| <b>Application Data Sheet 37 CFR 1.76</b> |                                                | Attorney Docket Number | LACEP001+ |
|                                           |                                                | Application Number     |           |
| Title of Invention                        | USING GRAPH-BASED MODELS TO IDENTIFY ANOMALIES |                        |           |

|                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                      |                      |                      |                                       |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|----------------------|----------------------|---------------------------------------|
| <b>Assignee</b>                                                                                                                                                                                                                                                                                                                                                                                                                                        | 1                    |                      |                      |                                       |
| Complete this section if assignee information, including non-applicant assignee information, is desired to be included on the patent application publication. An assignee-applicant identified in the "Applicant Information" section will appear on the patent application publication as an applicant. For an assignee-applicant, complete this section only if identification as an assignee is also desired on the patent application publication. |                      |                      |                      |                                       |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                      |                      |                      | <input type="button" value="Remove"/> |
| If the Assignee or Non-Applicant Assignee is an Organization check here. <input type="checkbox"/>                                                                                                                                                                                                                                                                                                                                                      |                      |                      |                      |                                       |
| Prefix                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Given Name           | Middle Name          | Family Name          | Suffix                                |
| <input type="text"/>                                                                                                                                                                                                                                                                                                                                                                                                                                   | <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="text"/>                  |
| <b>Mailing Address Information For Assignee including Non-Applicant Assignee:</b>                                                                                                                                                                                                                                                                                                                                                                      |                      |                      |                      |                                       |
| Address 1                                                                                                                                                                                                                                                                                                                                                                                                                                              |                      | <input type="text"/> |                      |                                       |
| Address 2                                                                                                                                                                                                                                                                                                                                                                                                                                              |                      | <input type="text"/> |                      |                                       |
| City                                                                                                                                                                                                                                                                                                                                                                                                                                                   | <input type="text"/> | State/Province       | <input type="text"/> |                                       |
| Country <sup>i</sup>                                                                                                                                                                                                                                                                                                                                                                                                                                   | <input type="text"/> | Postal Code          | <input type="text"/> |                                       |
| Phone Number                                                                                                                                                                                                                                                                                                                                                                                                                                           | <input type="text"/> | Fax Number           | <input type="text"/> |                                       |
| Email Address                                                                                                                                                                                                                                                                                                                                                                                                                                          | <input type="text"/> |                      |                      |                                       |
| Additional Assignee or Non-Applicant Assignee Data may be generated within this form by selecting the Add button.                                                                                                                                                                                                                                                                                                                                      |                      |                      |                      | <input type="button" value="Add"/>    |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                      |                      |                      |                                       |

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                |           |        |                                       |            |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------|--------|---------------------------------------|------------|
| <b>Signature:</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |           |        | <input type="button" value="Remove"/> |            |
| <b>NOTE:</b> This Application Data Sheet must be signed in accordance with 37 CFR 1.33(b). <b>However, if this Application Data Sheet is submitted with the INITIAL filing of the application and either box A or B is not checked in subsection 2 of the "Authorization or Opt-Out of Authorization to Permit Access" section, then this form must also be signed in accordance with 37 CFR 1.14(c).</b>                                                                                             |                |           |        |                                       |            |
| This Application Data Sheet <b>must</b> be signed by a patent practitioner if one or more of the applicants is a <b>juristic entity</b> (e.g., corporation or association). If the applicant is two or more joint inventors, this form must be signed by a patent practitioner, <b>all</b> joint inventors who are the applicant, or one or more joint inventor-applicants who have been given power of attorney (e.g., see USPTO Form PTO/AIA/81) on behalf of <b>all</b> joint inventor-applicants. |                |           |        |                                       |            |
| See 37 CFR 1.4(d) for the manner of making signatures and certifications.                                                                                                                                                                                                                                                                                                                                                                                                                             |                |           |        |                                       |            |
| Signature                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | /Robyn Wagner/ |           |        | Date (YYYY-MM-DD)                     | 2017-11-27 |
| First Name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Robyn          | Last Name | Wagner | Registration Number                   | 50575      |
| Additional Signature may be generated within this form by selecting the Add button.                                                                                                                                                                                                                                                                                                                                                                                                                   |                |           |        | <input type="button" value="Add"/>    |            |

|                                           |                                                |                        |           |
|-------------------------------------------|------------------------------------------------|------------------------|-----------|
| <b>Application Data Sheet 37 CFR 1.76</b> |                                                | Attorney Docket Number | LACEP001+ |
|                                           |                                                | Application Number     |           |
| Title of Invention                        | USING GRAPH-BASED MODELS TO IDENTIFY ANOMALIES |                        |           |

This collection of information is required by 37 CFR 1.76. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 23 minutes to complete, including gathering, preparing, and submitting the completed application data sheet form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

## Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether the Freedom of Information Act requires disclosure of these records.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

PROVISIONAL APPLICATION FOR  
UNITED STATES PATENT

**USING GRAPH-BASED MODELS TO IDENTIFY  
ANOMALIES**

By Inventors:

Vikram Kapoor  
Cupertino, CA

Rakesh Sachdeva  
Santa Clara, CA

Samuel Joseph Pullara III  
Los Altos, CA

Murat Bog  
Fremont, CA

Yijou Chen  
Cupertino, CA

Sanjay Kalra  
San Jose, CA

Harish Kumar Bharat Singh  
Mountain View, CA

Assignee: Lacework Inc.

VAN PELT, YI & JAMES LLP  
10050 N. Foothill Blvd., Suite 200  
Cupertino, CA 95014  
Telephone 408-973-2585

**[0001]** Figure 1 illustrates an example of an environment in which activities that occur within data centers are modeled. Using techniques described herein, a baseline of data center activity can be modeled, and deviations from that baseline can be identified as anomalous. Anomaly detection can be beneficial in a security context, and can also be useful in other contexts, such as devops.

**[0002]** Two example data centers (104 and 106) are shown in Figure 1, and belong to fictional companies ACME, and BETA, respectively. A data center can comprise dedicated equipment (e.g., owned and operated by ACME, or owned/leased by ACME and operated exclusively on ACME's behalf by a third party). A data center can also comprise cloud-based resources, such as infrastructure as a service (IaaS), platform as a service (PaaS), and/or software as a service (SaaS) elements. The techniques described herein can be used in conjunction with multiple types of data centers, including ones wholly using dedicated equipment, ones that are entirely cloud-based, and ones that use a mixture of both dedicated equipment and cloud-based resources.

**[0003]** Both data center 104 and data center 106 include a plurality of nodes, depicted collectively as set of nodes 108 and set of nodes 110, respectively, in Figure 1. Installed on each of the nodes are in-server / in-virtual machine (VM) agents (as applicable), which are configured to collect data and report it to platform 102 for analysis. Agents are small, self-contained binaries that can be run on any appropriate platforms, including virtualized ones (and, as applicable, within containers). Agents monitor the nodes on which they execute for a variety of different activities, including: connection, process, user, machine, and file activities. Agents can be executed in user space, and can use a variety of kernel modules (e.g., auditd, iptables, netfilter, pcap, etc.) to collect data. Agents can be implemented in any appropriate programming language, such as C or Golang, using applicable kernel APIs.

**[0004]** As will be described in more detail below, agents can selectively report information to platform 102 in varying amounts of detail and/or with variable frequency. As will also be described in more detail below, the data collected by agents is used by platform 102 to create polygraphs. In some embodiments, agents report information directly to platform 102. In other embodiments, at least some agents provide information to a data aggregator, such as data

aggregator 114, which in turn provides information to platform 102. The functionality of a data aggregator can be implemented as a separate binary or other application (distinct from an agent binary), and can also be implemented by having an agent execute in an “aggregator mode” in which the designated aggregator node acts as a Layer 7 proxy for other agents that do not have access to platform 102. Further, a chain of multiple aggregators can be used, if applicable (e.g., with agent 112 providing data to data aggregator 114, which in turn provides data to another aggregator (not pictured) which provides data to platform 102. An example way to implement an aggregator is through a program written in an appropriate language, such as C or Golang.

**[0005]** Use of an aggregator can be beneficial in sensitive environments (e.g., involving financial or medical transactions) where various nodes are subject to regulatory or other architectural requirements (e.g., prohibiting a given node from communicating with systems outside of data center 104). And, use of an aggregator can help to minimize security exposure more generally. As one example, by limiting communications with platform 102 to data aggregator 114, individual nodes in group 108 need not make external network connections (e.g., via Internet 124), which can potentially expose them to compromise (e.g., by other external devices, such as device 118, operated by a criminal). Similarly, platform 102 can provide updates, configuration information, etc., to data aggregator 114 (which in turn distributes them to nodes 108), rather than requiring nodes 108 to allow incoming connections from platform 102 directly.

**[0006]** Another benefit of an aggregator model is that network congestion can be reduced (e.g., with a single connection being made at any given time between data aggregator 114 and platform 102, rather than potentially many different connections being open between nodes 108 and platform 102). Similarly, network consumption can also be reduced (e.g., where the aggregator is able to apply compression techniques when bundling data received from multiple agents).

**[0007]** One example way that an agent (e.g., agent 112, installed on node 116) can provide information to data aggregator 114 is via a REST API, formatted using Apache Avro. One example type of information sent by agent 112 to data aggregator 114 is status information. Status information is sent periodically (e.g., once a minute) and includes (in various

embodiments): a. an amount of event backlog (in bytes) that has not yet been transmitted, b. configuration information, c. any data loss period for which data was dropped, d. a cumulative count of errors encountered since the agent started, e. version information for the agent binary, and f. cumulative statistics on data collection (e.g., number of network packets processed, new processes seen, etc.). A second example type of information sent by agent 112 to data aggregator 114 is event data, which is provided (in various embodiments) and includes a UTC timestamp for each event. As applicable, the agent can control the amount of data that it sends to the data aggregator in each call (e.g., a maximum of 10MB) by adjusting the amount of data sent to manage the conflicting goals of transmitting data as soon as possible, and maximizing throughput.

**[0008]** Each data aggregator runs within a customer environment. A data aggregator (e.g., data aggregator 114) facilitates data routing from many different agents (e.g., agents executing on nodes 108) to platform 102. In various embodiments, data aggregator 114 implements a SOCKS 5 caching proxy through which agents can connect to platform 102. As applicable, data aggregator 114 can encrypt (or otherwise obfuscate) sensitive information prior to transmitting it to platform 102, and can also distribute key material to agents which can encrypt the information (as applicable). Data aggregator 114 includes a local storage, to which agents can upload data (e.g., pcap packets). The storage has a key-value interface. The local storage can also be omitted, and agents configured to upload data to a cloud storage or other storage area, as applicable.

## **[0009] I. AGENT DATA COLLECTION AND REPORTING**

### **[0010] A. Initial Configuration**

**[0011]** In the following example, suppose that a network administrator at ACME (hereinafter “Alice”) has decided to begin using the services of platform 102. In some embodiments, Alice accesses a web frontend (e.g., web app 120), using her computer (126) and enrolls (on behalf of ACME) an account with platform 102. After enrollment is complete, Alice is presented with a set of installers, pre-built and customized for the ACME environment, that she can download from platform 102 and deploy on nodes 108. Examples of such installers include: a Windows executable file; an iOS app; and a Linux package (e.g., .deb or .rpm), binary,

or Docker container. When a network administrator at BETA (hereinafter “Bob”) also signs up for the services of platform 102, he will similarly be presented with a set of installers that are pre-built and customized for the BETA environment.

**[0012]** Alice deploys an appropriate installer on each of nodes 108 (e.g., with a Windows executable file deployed on a Windows-based platform, and a Linux package deployed on a Linux platform, as applicable). As applicable, the agent can be deployed in a container (e.g., using a swarm). Agent deployment can also be performed using one or more appropriate automation tools, such as Chef and Puppet.

**[0013]** In various embodiments, the agent is installed in the user space (i.e., is not a kernel module), and the same binary is executed on each node of the same type (e.g., all Windows-based platforms have the same Windows-based binary installed on them). The primary job of an agent, such as agent 112, is to collect data (e.g., associated with node 116) and report it (e.g., to data aggregator 114). Other tasks that can be performed by agents include data configuration and upgrading.

**[0014] B. Data Collection**

**[0015]** One approach to collecting data is to collect virtually all information available about a node (and, e.g., the processes running on it). A downside of this approach is that collecting information can be resource intensive (i.e., requiring potentially large amounts of processor, network, and storage resources). Another downside of this approach is that it can result in the collection of private or other sensitive information that can be difficult to filter/exclude from downstream reporting. An alternate approach is for the agent to monitor for network connections, and then begin collecting information about those processes associated with the network connections, using the presence of a network packet associated with a process as a trigger for collecting additional information about the process. As an example, if a user of node 116 executes a calculator application, no information about use of that application is collected by agent 112/sent to data aggregator 114. If, however, the user of node 116 executes an ssh command (e.g., to ssh from node 116 to node 122), agent 112 will collect information about the process and provide associated information to data aggregator 114. In various

embodiments, the agent always collects/reports information about certain events, such as privilege escalation, irrespective of whether the event is associated with network activity.

**[0016]** An approach to collecting information (e.g., by an agent) is as follows, and described in conjunction with process 200 depicted in Figure 2. An agent (e.g., agent 112) monitors its node (e.g., node 116) for network activity. One example way that agent 112 can monitor node 116 for network activity is by listening using libpcap. As packets are received (202), the agent obtains and maintains (e.g., in an in memory cache) connection information associated with the network activity (204). Examples of such information include DNS query/response, TCP, and UDP information.

**[0017]** The agent also determines a process associated with the network connection (206). One example approach is for the agent to use netlink\_diag to obtain inode/process information from the kernel. Another example approach is for the agent to scan using netstat (e.g., on /proc/net/tcp, /proc/net/tcp6, /proc/net/udp, and /proc/net/udp6) to obtain sockets and relate them to processes. Information such as socket state (e.g., whether a socket is connected, listening, etc.) can also be collected by the agent.

**[0018]** One way an agent can obtain a mapping between a given inode and a process identifier is to scan within the /proc/pid directory. For each of the processes currently running, the agent examines each of their file descriptors. If a file descriptor is a match for the inode, the agent can determine that the process associated with the file descriptor owns the inode. Once a mapping is determined between an inode and a process identifier, the mapping is cached. As additional packets are received for the connection, the cached process information is used (rather than a new search being performed).

**[0019]** In some cases, exhaustively scanning for an inode match across every file descriptor may not be feasible (e.g., due to CPU limitations). In various embodiments, the searching through file descriptors is accordingly optimized. User filtering is one example of such an optimization. A given socket is owned by a user. Any processes associated with the socket will be owned by the same user as the socket. When matching an inode (identified as relating to a given socket) against processes, the agent can filter through the processes and only examine the file descriptors of processes sharing the same user owner as the socket. In various

embodiments, processes owned by root are always searched against (e.g., even when user filtering is employed).

**[0020]** Another example of an optimization is to prioritize searching the file descriptors of certain processes over others. One such prioritization is to search through the subdirectories of /proc/ starting with the youngest process. One approximation of such a sort order is to search through /proc in reverse order (e.g., examining highest numbered processes first). Higher numbered processes are more likely to be newer (i.e., not long-standing processes), and thus more likely to be associated with new connections (i.e., ones for which inode-process mappings are not already cached). In some cases, the most recently created process may not have the highest process identifier (e.g., due to the kernel wrapping through process identifiers).

**[0021]** Another example prioritization is to query the kernel for an identification of the most recently created process and to search in a backward order through the directories in /proc (e.g., starting at process 400 and working backwards, then wrapping to the highest value (e.g., 32768) and continuing to work backward from there). An alternate approach is for the agent to keep track of the newest process that it has reported information on (e.g., to data aggregator 114), and begin its search of /proc in a forward order starting from the PID of that process.

**[0022]** Another example prioritization is to maintain, for each user actively using node 116, a list of the five most recently active processes. Those processes are more likely than other processes (less active, or passive) on node 116 to be involved with new connections, and can thus be searched first. For many processes, lower valued file descriptors tend to correspond to non-sockets (e.g., stdin, stdout, stderr). Yet another optimization is to preferentially search higher valued file descriptors (e.g., across processes) over lower valued file descriptors (that are less likely to yield matches).

**[0023]** In some cases, while attempting to locate a process identifier for a given inode, an agent may encounter a socket that does not correspond to the inode being matched against and is not already cached. The identity of that socket (and its corresponding inode) can be cached, once discovered, thus removing a future need to search for that pair.

**[0024]** In some cases, a connection may terminate before the agent is able to determine its associated process (e.g., due to a very short-lived connection (100ms), due to a backlog in agent processing, etc.). One approach to addressing such a situation is to asynchronously collect information about the connection using the audit kernel API, which streams information to user space. The information collected from the audit API (which can include PID/inode information) can be matched by the agent against pcap/inode information. In some embodiments, the audit API is always used, for all connections. However, due to CPU utilization considerations, use of the audit API can also be reserved for short/otherwise problematic connections (and/or omitted, as applicable).

**[0025]** Once the agent has determined which process is associated with the network connection (206), the agent can then collect additional information associated with the process (208). As will be described in more detail below, some of the collected information comprises attributes of the process (e.g., a process parent hierarchy, and an identification of a binary associated with the process). As will also be described in more detail below, other of the collected information is derived (e.g., session summarization data and hash values).

**[0026]** The collected information is then transmitted (210), e.g., by an agent (e.g., agent 112) to a data aggregator (e.g., data aggregator 114), which in turn provides it to platform 102. In some embodiments, all information collected by an agent is transmitted (e.g., to a data aggregator and/or to platform 102). In other embodiments, the amount of data transmitted is minimized (e.g., for efficiency reasons), using various techniques.

**[0027]** One approach to minimizing the amount of data flowing from agents (such as agents installed on nodes 108) to platform 102 is to use a technique of implicit references with unique keys. The keys can be explicitly used by platform 102 to extract/derive relationships, as necessary, in a data set at a later time, without impacting performance.

**[0028]** Depicted in Figure 3A is initial information collected by an agent about a process. Region 302 indicates the time at which the agent generated event 300. Region 304 indicates that event 300 pertains to a process. The type identifies a physical collection of related attributes. Other examples of event types that the agent can collect and report (and examples of the types of information included in such reports) are provided below (see Additional Agent Data Examples).

**[0029]** Region 306 indicates the process ID (PID) of the process, and region 310 indicates the time the process was started. In region 308 is depicted a hash, generated by the agent to uniquely identify the particular process being reported on in event 300 (e.g., among all other processes whose data is reported to platform 102). In various embodiments, such hashes (and other hashes, such as are shown at 312-316, are generated pseudo-randomly. The hashes can be reconstructed in the event the agent crashes and restarts. An example salt for the hashes is a combination of a local process identifier, start time of the process, and one or more machine attributes.

**[0030]** Region 318 indicates whether the process is associated with a terminal, which generally indicates whether a process is interactive. Region 320 indicates a SHA-256 hash of the entire command line used to start the process, including any variable arguments, and region 322 indicates the path to the executable. Region 324 indicates the effective username of the owner of the process (which can either be the same or different from the username shown in region 328). For example, suppose a user logs into node 116 using a first username, and then becomes a different user (e.g., using the `setuid` system call to inherit the permissions of a second user), both names can be preserved in regions 324 and 328, respectively. Finally, region 326 indicates whether the process is running as a container isolation.

**[0031]** As previously mentioned, some data collected about a process is constant and does not change over the lifetime of the process (e.g., attributes), and some data changes (e.g., statistical information and other variable information). The data depicted in Figure 3A is an example of constant data. It can be transmitted (210) once, when the agent first becomes aware of the process. And, if any changes to the data depicted in Figure 3A are detected (e.g., a process changes its parent), a refreshed version of the data depicted in Figure 3A can be transmitted (210) as applicable.

**[0032]** Figure 3B illustrates an example of variable data that can be collected using the process shown in Figure 2. Variable data such as is shown in Figure 3B can be transmitted (210) at periodic (or other) intervals. Region 352 indicates a timestamp of when dataset 350 was collected by the agent. The hash depicted in region 354 is the same hash that was included in region 308, and is used within platform 102 to join process creation time attributes (e.g., shown

in Figure 3A) with runtime attributes (e.g., shown in Figure 3B) to construct a full dataset. Region 356 indicates a thread count for the process. Region 358 indicates the total virtual memory used by the process. Region 360 indicates the total resident memory used by the process. Region 362 indicates the total time spent by the process executing in user space. Region 364 indicates the total time spent by the process executing in kernel space.

**[0033] 1. Additional Agent Data Examples**

**[0034]** Below are additional examples of data that an agent, such as agent 112, can collect and provide to platform 102.

**[0035] a. User Data**

**[0036]** Core User Data: user name, UID (user ID), primary group, other groups, home directory.

**[0037]** Failed Login Data: IP address, hostname, username, count.

**[0038]** User Login Data: user name, hostname, IP address, start time, TTY (terminal), UID (user ID), GID (group ID), process, end time.

**[0039] b. Machine Data**

**[0040]** Dropped Packet Data: source IP address, destination IP address, destination port, protocol, count.

**[0041]** Machine Data: hostname, domain name, architecture, kernel, kernel release, kernel version, OS, OS version, OS description, CPU, memory, model number, number of cores, last boot time, last boot reason, (AWS) tags, default router, interface name, interface hardware address, interface IP address and mask, promiscuous mode.

**[0042] c. Network Data**

**[0043]** Network Connection Data: source IP address, destination IP address, source port, destination port, protocol, start time, end time, incoming and outgoing bytes, source process, destination process, direction of connection.

**[0044] d. User Data**

**[0045]** Listening Ports in Server: source IP address, port number, protocol, process.

**[0046] e. User Data**

**[0047]** Dropped Packet Data: source IP address, destination IP address, destination port, protocol, count.

**[0048]** Arp Data: source hardware address, source IP address, destination hardware address, destination IP address.

**[0049]** DNS Data: source IP address, response code, response string, question (request), packet length, final answer (response).

**[0050] f. Application Data**

**[0051]** Package Data: exe path, package name, architecture, version, package path, checksums (MD5, SHA-1, SHA-256), size, owner, owner ID.

**[0052]** Application Data: command line, PID (process ID), start time, UID (user ID), EUID (effective UID), PPID (parent process ID), PGID (process group ID), SID (session ID), exe path, username, container ID.

**[0053] g. Container Data**

**[0054]** Container Image Data: image creation time, parent ID, author, container type, repo, (AWS) tags, size, virtual size, image version.

**[0055]** Container Data: container start time, container type, container name, container ID, network mode, privileged, PID mode, IP addresses, listening ports, volume map, process ID.

**[0056] 2. Additional Information About Container Usage**

**[0057]** As mentioned above, an agent, such as agent 112 can be deployed in a container (e.g., a Docker container), and can also be used to collect information about containers. Collection about a container can be performed by an agent irrespective of whether the agent is

itself deployed in a container or not (as the agent can be deployed in a container running in a privileged mode that allows for monitoring).

**[0058]** Agents can discover containers (e.g., for monitoring) by listening for container create events (e.g., provided by Docker), and can also perform periodic ordered discovery scans to determine whether containers are running on a node. When a container is discovered, the agent can obtain attributes of the container, e.g., using standard Docker API calls (e.g., to obtain IP address associated with container, whether there's a server running inside, what port it is listening on, associated PIDs, etc.). Information such as the parent process that started the container can also be collected, as can information about the image (which comes from the Docker repository).

**[0059]** In various embodiments, agents use namespaces to determine whether a process is associated with a container or not. Namespaces are a feature of the Linux kernel that can be used to isolate resources of a collection of processes. Examples of namespaces include process ID (PID) namespaces, network namespaces, and user namespaces. Given a process, the agent can perform a fast lookup to determine whether the process is part of the namespace the container claims to be its namespace.

**[0060]** **C. Data Reporting**

**[0061]** As mentioned above, agents can be configured to report certain types of information (e.g., attribute information) once, when the agent first becomes aware of a process. In various embodiments, such static information is not reported again (or is reported once a day or every twelve hours), unless it changes (e.g., a process changes its parent, changes its owner, or a SHA-1 of the binary associated with the process changes).

**[0062]** In contrast to static/attribute information, certain types of data changes constantly (e.g., network-related data). In various embodiments, agents are configured to report a list of current connections every minute (or other appropriate time interval). In that connection list will be connections that started in that minute interval, connections that ended in that minute interval, and connections that were ongoing throughout the minute interval (e.g., a one minute slice of a one hour connection).

**[0063]** In various embodiments, agents are configured to collect/compute statistical information about connections (e.g., at the one minute level of granularity). Examples of such information include, for the time interval, the number of bytes transferred, and in which direction. Another example of information collected by an agent about a connection is the length of time between packets. For connections that span multiple time intervals (e.g., a seven minute connection), statistics are calculated for each minute of the connection. Such statistical information (for all connections) can be reported (e.g., to a data aggregator) once a minute.

**[0064]** In various embodiments, agents are also configured to maintain histogram data for a given network connection, and provide it (in the Apache Avro data exchange) under the Connection event type data. Examples of such histograms include: 1. a packet length histogram (packet\_len\_hist), which characterizes network packet distribution; 2. a session length histogram (session\_len\_hist), which characterizes a network session length; 3. a session time histogram (session\_time\_hist), which characterizes a network session time; and 4. a session switch time histogram (session\_switch\_time\_hist), which characterizes network session switch time (i.e., incoming->outgoing and vice versa). Figures 3C and 3D depict, side-by-side, examples of full outgoing (3C) and incoming (3D) network data sets, including each of the four histograms. Each of the histograms shown in Figures 3C and 3D includes the following fields: 1. count, which provides a count of the elements in the sampling; 2. sum, which provides a sum of elements in the sampling; 3. max, which provides the highest value element in the sampling; 4. std\_dev, which provides the standard deviation of elements in the sampling; and 5. buckets, which provides a discrete sample bucket distribution of sampling data (if applicable).

**[0065]** For some protocols (e.g., HTTP), typically, a connection is opened, a string is sent, a string is received, and the connection is closed. For other protocols (e.g., NFS), both sides of the connection engage in a constant chatter. Histograms allow platform 102 to model application behavior (e.g., using machine learning techniques), for establishing baselines, and for detecting deviations. As one example, suppose that a given HTTP server typically sends/receives 1,000 bytes (in each direction) whenever a connection is made with it. If a connection generates 500 bytes of traffic, or 2,000 bytes of traffic, such connections would be considered within the typical usage pattern of the server. Suppose, however, that a connection is

made that results in 10G of traffic. Such a connection is anomalous and can be flagged accordingly.

**[0066] D. Storing Data Collected by Agents**

**[0067]** Returning to Figure 1, as previously mentioned, data aggregator 114 is configured to provide information (e.g., collected from nodes 108 by agents) to platform 102. And, data aggregator 128 is similarly configured to provide information to platform 102. As shown in Figure 1, both aggregator 114 and aggregator 128 connect to a load balancer 130, which accepts connections from aggregators (and/or, as applicable agents), as well as other devices, such as computer 126 (e.g., when it communicates with web app 120), and supports fair balancing. In various embodiments, load balancer 130 is a reverse proxy that load balances accepted connections internally to various microservices (described in more detail below), allowing for services provided by platform 102 to scale up as more agents are added to the environment and/or more entities subscribe to services provided by platform 102. Example ways to implement load balancer 130 include using HaProxy, using nginx, and using elastic load balancing (ELB) services made available by Amazon.

**[0068]** Agent service 132 is a microservice that is responsible for accepting data collected from agents (e.g., provided by aggregator 114). In various embodiments, agent service 132 uses a standard secure protocol, such as HTTPS to communicate with aggregators (and as applicable agents), and receives data in an appropriate format such as Apache Avro). When agent service 132 receives an incoming connection, it can perform a variety of checks, such as whether the data is being provided by a current customer, and whether the data is being provided in an appropriate format. If the data is not appropriately formatted (or is not provided by a current customer), it is rejected. If the data is appropriately formatted, agent service 132 facilitates copying the received data to a stable storage (e.g., using Amazon Kinesis (134)). Once the ingesting into Kinesis is complete, service 132 sends an acknowledgement to the data provider (e.g., data aggregator 114). If the agent does not receive such an acknowledgement, it is configured to retry sending the data to platform 102. One way to implement agent service 132 is as a REST API server framework (e.g., Java DropWizard), configured to communicate with Kinesis (e.g., using a Kinesis library).

**[0069]** In various embodiments, platform 102 uses a single Kinesis stream (134) for all incoming customer data (e.g., including data provided by data aggregator 114 and data aggregator 128), and the data is sharded based on the node (also referred to herein as a “machine”) that originated the data (e.g., node 116 vs. node 122), with each node having a globally unique identifier within platform 102. Multiple instances of agent service 132 can write to multiple shards.

**[0070]** Kinesis has a retention period of one day. To persist data longer than a day, it is copied to long term storage (e.g., S3). S3 Loader 136 is a microservice that is responsible for picking up data from Kinesis stream 134 and persisting it in S3 (138). In one example embodiment, files collected by S3 Loader 136 from the Kinesis stream are placed into a single bucket, but segmented using a combination of a customer identifier and time slice. Given a particular time segment, and a given customer identifier, the corresponding file (stored in S3) contains five minutes’ (or another appropriate time slice) of data collected at that specific customer from all of the customer’s nodes. S3 Loader 136 can be implemented in any appropriate programming language, such as Java or C, and can be configured to use a Kinesis library to interface with Kinesis. In various embodiments, S3 Loader 136 uses the Amazon Simple Queue Service (SQS) (e.g., to alert DB Loader 140 that there is work for it to do).

**[0071]** DB Loader 140 is a microservice that is responsible for loading data into an appropriate database service (142), such as SnowflakeDB or Amazon Redshift, using individual per-customer databases. In particular, DB Loader 140 is configured to periodically load data into a set of raw tables from files created by S3 Loader as per above. DB Loader 140 manages throughput, errors, etc., to make sure that data is loaded consistently and continuously. Further, DB Loader 140 can read incoming data and load into database 142 data that is not already present in database 142’s tables. DB Loader 140 can be implemented in any appropriate programming language, such as Java or C, and an SQL framework such as jOOQ (e.g., to manage SQLs for insertion of data), and SQL/JDBC libraries. DB Loader 140 also uses Amazon S3 and Amazon Simple Queue Service (SQS) to manage files being transferred to and from the database (e.g., Snowflake).

**[0072]** Customer data included in database 142 can be augmented with data from additional data sources, such as AWS Cloud Trail Analyzer 144, which is another microservice. AWS Cloud Trail Analyzer 144 pulls data using Amazon Cloudtrail for each applicable customer account, as soon as the data is available. AWS Cloud Trail Analyzer 144 normalizes the Cloudtrail data as applicable, so that it can be inserted into database 142 for later querying/analysis. AWS Cloud Trail Analyzer 144 can be written in any appropriate programming language, such as Java or C. AWS Cloud Trail Analyzer 144 also makes use of SQL/JDBC libraries to interact with database 142 to insert/query data.

**[0073] II. POLYGRAPHS**

**[0074] A. Polygraph Overview**

**[0075]** As previously mentioned, platform 102 can model activities that occur within data centers, such as data centers 104 and 106. The model is stable over time, and even subtle differences (e.g., between a current state of the data center and the model) can be surfaced.

**[0076]** Using techniques described herein, platform 102 can automatically discover entities deployed in a given data center. Examples of entities include workloads, applications, processes, machines, virtual machines, containers, files, and users. The entities are grouped together logically (into analysis groups) based on behaviors, and temporal behavior baselines can be established. In particular, using techniques described herein, hourly graphs can be constructed (also referred to herein as polygraphs), in which the nodes are applicable logical entities, and the edges represent behavioral relationships between the logical entities in the graph. Baselines can be created for every node and edge.

**[0077]** As mentioned above, communication (e.g., between applications/nodes) is an example of a behavior. A model of communications between processes is an example of a behavioral model. As another example, the launching of applications is another example of a behavior that can be modeled. The baselines are updated hourly for every entity. Deviations from the expected normal behavior can then be detected and automatically reported (e.g., as anomalies or threats detected). Such deviations may be due to a desired change, a

misconfiguration, or malicious activity. As applicable, platform 102 can score the detected deviations (e.g., based on severity and threat posed).

**[0078]** In addition to models of application/process communications and models of application launches, additional examples of analysis groups include models of machine communications, models of privilege changes, and models of insider behaviors (monitoring the interactive behavior of human users as they operate within the data center).

**[0079]** Two types of information collected by agents (described in more detail above) are network level information and process level information. As previously mentioned, agents collect information about every connection involving their respective nodes. And, for each connection, information about both the server and the client is collected (e.g., using the connection-to-process identification techniques described above). DNS queries and responses are also collected. Examples of process level information collected by agents include attributes (user ID, effective user ID, and command line). Information such as what user/application is responsible for launching a given process, the binary being executed (and its SHA-256 values) is also provided by agents.

**[0080]** The dataset collected by agents across a datacenter can be very large, and many resources (e.g., virtual machines, IP addresses, etc.) are recycled very quickly. For example, an IP address and port number used at a first point in time by a first process on a first virtual machine may very rapidly be used (e.g., an hour later) by a different process/virtual machine.

**[0081]** A dataset (and elements within it) can be considered at both a physical level, and a logical level, as illustrated in Figure 4. In particular, Figure 4 illustrates an example 5-tuple of data (402) collected by an agent, represented physically (414) and logically (416). The 5-tuple includes a source address (404), a source port (406), a destination address (408), a destination port (410), and a protocol (412). In some cases, port numbers (e.g., 406, 410) may be indicative of the nature of a connection (e.g., with certain port usage standardized). However, in many cases, and in particular in data centers, port usage is ephemeral. For example, a Docker container can listen on an ephemeral port, which is unrelated to the service it will run. When another Docker container starts (for the same service), the port may well be different. Similarly,

particularly in a virtualized environment, IP addresses may be recycled frequently (and are thus also potentially ephemeral).

**[0082]** A physical representation of the 5-tuple is depicted in region 414. A process 418 (executing on machine 420) has opened a connection to machine 422. In particular, process 418 is in communication with process 424. Information such as the number of packets exchanged between the two machines over the respective ports can be recorded.

**[0083]** As previously mentioned, in a data center environment, portions of the 5-tuple may change – potentially frequently – but still be associated with the same behavior. Namely, one application (e.g., Apache), may frequently be in communication with another application (e.g., Oracle), using ephemeral data center resources. Further, either/both of Apache and Oracle may be multi-homed. This can lead to potentially thousands of 5-tuples (or more) that all correspond to Apache communicating with Oracle within a data center. For example, Apache could be executed on a single machine, and could also be executed across fifty machines, which are variously spun up and down (with different IP addresses each time). An alternate representation of the 5-tuple shown in region 402 is depicted in region 416, and is logical. The logical representation of the 5-tuple aggregates the 5-tuple (along with other connections between Apache and Oracle having other 5-tuples) as logically representing the same connection. By aggregating data from raw physical connection information into logical connection information, using techniques described herein, a size reduction of six orders of magnitude in the data set can be achieved.

**[0084]** Figure 5 depicts a portion of a logical polygraph. In the example shown, a data center has seven instances of the application `update_engine` (502), executing as seven different processes on seven different machines, having seven different IP addresses, and using seven different ports. The instances of `update_engine` variously communicate with `update.core-os.net` (504), which may have a single IP address or many IP addresses itself, over the one hour time period reflected in the polygraph. In the example shown in Figure 5, `update_engine` is a client, connecting to `update.core-os.net`, as indicated by arrow 508.

**[0085]** Behaviors of the seven processes are clustered together, into a single summary. As indicated in region 506, statistical information about the connections is also maintained (e.g.,

number of connections, histogram information, etc.). A polygraph such as is depicted in Figure 5 can be used to establish a baseline of behavior (e.g., at the one-hour level), allowing for the future detection of deviations from that baseline. As one example, suppose that statistically an `update_engine` instance transmits data at 11 bytes per second. If an instance were instead to transmit data at 1000 bytes per second, such behavior would represent a deviation from the baseline and could be flagged accordingly. Similarly, changes that are within the baseline (e.g., an eighth instance of `update_engine` appears, but otherwise behaves as the other instances; or one of the seven instances disappears) are not flagged as anomalous. Further, data center events, such as failover, autobalancing, and A-B refresh are unlikely to trigger false alarms in a polygraph, as at the logical level, the behaviors remain the same.

**[0086]** In various embodiments, polygraphs are maintained for every application in a data center, and such polygraphs can be combined to make a single data center view across all such applications. Figure 6 illustrates a portion of a polygraph for a service that evidences more complex behaviors than are depicted in Figure 5. In particular, Figure 6 illustrates the behaviors of S3 as a service (as used by a particular customer data center). Clients within the datacenter variously connect to the S3 service using one of five fully qualified domains (listed in region 602). Contact with any of the domains is aggregated as contact with S3 (as indicated in region 604). Depicted in region 606 are various containers which (as clients) connect with S3. Other containers (which do not connect with S3) are not included in the polygraph. As with the polygraph depicted in Figure 5, statistical information about the connections is known and summarized, such as the number of bytes transferred, histogram information, etc.

**[0087]** Figure 7 illustrates a polygraph for a data center. In particular, the polygraph indicates a one hour summary of approximately 500 virtual machines, which collectively run one million processes, and make 100 million connections in that hour. As illustrated in Figure 7, a polygraph represents a drastic reduction in size (e.g., from tracking information on 100 million connections in an hour, to a few hundred nodes and a few hundred edges). Further, as a data center scales up (e.g., from using 10 virtual machines to 100 virtual machines as the data center uses more workers to support existing applications) the polygraph for the data center will tend to stay the same size (with the 100 virtual machines clustering into the same nodes that the 10

virtual machines previously clustered into). As new applications are added into the data center, the polygraph will automatically grow to include behaviors involving those applications.

**[0088]** In the particular polygraph shown in Figure 7, nodes generally correspond to workers, and edges correspond to communications the workers engage in. The graphed communications include traffic entering the data center, traffic exiting the data center, and traffic that stays wholly within the data center (e.g., traffic between workers). One example of a node included in polygraph 700 is the sshd application, depicted as node 702. As indicated in Figure 7, 421 instances of sshd were executing during the one hour time period of data represented in polygraph 700. As indicated in region 704, nodes within the data center communicated with a total of 1349 IP addresses outside of the data center (and not otherwise accounted for, e.g., as belonging to a service such as Amazon AWS (706) or Slack (708)).

**[0089] B. Interacting with Polygraphs**

**[0090]** In the following examples, suppose that Bob, an administrator of data center 106, is interacting with platform 102 to view visualizations of polygraphs in a web browser (e.g., as served to him via web app 120). One type of polygraph Bob can view is an application-communication polygraph, which indicates, for a given one hour window, which applications communicated with which other applications. Another type of polygraph Bob can view is an application launch polygraph. Bob can also view graphs related to user behavior, such as an insider user graph which tracks user connections (e.g., to internal and external applications, including chains of such behavior), a privilege change graph which tracks how privileges change between processes, and a user login polygraph, which tracks which (logical) machines a user logs into.

**[0091]** Figure 8 illustrates an example of an application-communication polygraph for a data center (e.g., data center 106) for the one hour period of 9am-10am on June 5. The time slice currently being viewed is indicated in region 802. If Bob clicks his mouse in region 804, Bob will be shown a representation of the application-communication polygraph as generated for the following hour (10am-11am on June 5).

**[0092]** Figure 9 depicts what is shown in Bob's browser after he has clicked on region 804, and has further clicked on region 806. The selection in region 806 turns on and off the ability to compare two time intervals to one another. Bob can select from a variety of options when comparing the 9am-10am and 10am-11am time intervals. By clicking region 904, Bob will be shown the union of both graphs (i.e., any connections that were present in either time interval). By clicking region 906, Bob will be shown the intersection of both graphs (i.e., only those connections that were present in both time intervals).

**[0093]** As shown in Figure 9, Bob has elected to click on region 908, which depicts connections that are only present in the 9am-10am polygraph in purple (910), and depicts connections that are only present in the 10am-11am polygraph in blue (912). Connections present in both polygraphs are omitted from display. As one example, in the 9am-10am polygraph (corresponding to connections made during the 9am-10am time period at data center 106), a connection was made by a server to sshd (914) and also to systemd (916). Both of those connections ended prior to 10am and are thus depicted in purple. As another example, in the 10am-11am polygraph (corresponding to connections made during the 10am-11am time period at data center 106), a connection was made from a known bad external IP to nginx (918). The connection was not present during the 9am-10am time slice and thus is depicted in blue. As yet another example, two different connections were made to a Slack service between 9am and 11am. However, the first was made by a first client during the 9am-10am time slice (920) and the second was made by a different client during the 10am-11am slice (922), and so the two connections are depicted respectively in purple and blue.

**[0094]** Returning to the polygraph depicted in Figure 8, suppose Bob enters "etcd" into the search box located in region 810. Bob will then be presented with the interface illustrated in Figure 10. As shown in Figure 10, three applications containing the term "etcd" were engaged in communications during the 9am-10am window. One application is etcdctl, a command line client for etcd. As shown in Figure 10, a total of three different etcdctl processes were executed during the 9am-10am window, and were clustered together (1002). Figure 10 also depicts two different clusters that are both named etcd2. The first cluster includes (for the 9am-10am window) five members (1004) and the second cluster includes (for the same window) eight members (1006). The reason for these two distinct clusters is that the two groups of applications

behave differently (e.g., they exhibit two distinct sets of communication patterns). Specifically, the instances of etcd2 in cluster 1004 only communicate with locksmithctl (1008) and other etcd2 instances (in both clusters 1004 and 1006). The instances of etcd2 in cluster 1006 communicate with additional entities, such as etcdctl and Docker containers. As desired, Bob can click on one of the clusters (e.g., cluster 1004) and be presented with summary information about the applications included in the cluster, as is shown in Figure 11 (e.g., in region 1102). He can also double click on a given cluster (e.g., cluster 1004) to see details on each of the individual members of the cluster broken out.

**[0095]** Suppose Bob now clicks on region 812 of the interface shown in Figure 8. Bob will then be shown an application launch polygraph. Launching an application is another example of a behavior. The launch polygraph models how applications are launched by other applications. Figure 12 illustrates an example of a portion of a launch polygraph. In particular, Bob has typed “find” into region 1202, to see how the “find” application is being launched. As shown in Figure 12, in the launch polygraph for the 10am-11am time period, find applications (1204) are always launched by bash (1206), which is in turn always launched by systemd (1208). If find is launched by a different application, this would be anomalous behavior.

**[0096]** Figure 13 illustrates another example of a portion of an application launch polygraph. In Figure 13, Bob has searched (1302) for “python ma” to see how “python marathon\_lb” is launched. As shown in Figure 13, in each case (during the one hour time slice of 10am-11am), python marathon\_lb is launched as a result of a chain of the same seven applications each time. If python marathon\_lb is ever launched in a different manner, this indicates anomalous behavior. The behavior could be indicative of malicious activities, but could also be due to other reasons, such as a misconfiguration, a performance-related issue, and/or a failure, etc.

**[0097]** Suppose Bob now clicks on region 814 of the interface shown in Figure 8. Bob will then be shown an inside user graph. The inside user graph tracks information about behaviors such as processes started by a user, and any processes started by those processes. As one example, suppose an administrator logs into a first virtual machine in data center 106 (e.g., using sshd via an external connection he makes from a hotel), using a first set of credentials (e.g.,

charlie.smith@example.com and an appropriate password). From the first virtual machine, the administrator connects to a second virtual machine (e.g., using the same credentials), then uses the sudo command to change identities to those of another user, and then launches a program. Graphs built by platform 102 can be used to associate the administrator with each of his actions, including launching the program using the identity of another user.

**[0098]** Figure 14 illustrates an example of a portion of an inside user behavior graph. In particular, in Figure 14, Bob is viewing a graph that corresponds to the time slice of 3pm-4pm on June 1. Figure 14 illustrates the internal/external applications that users connected to during the one hour time slice. If a user typically communicates with particular applications, that information will become part of a baseline. If the user deviates from his baseline behavior (e.g., using new applications, or changing privilege in anomalous ways), such anomalies can be surfaced.

**[0099]** Figure 15 illustrates an example of a portion of a privilege change graph, which identifies how privileges are changed between processes. Typically, when a user launches a process (e.g., “ls”), the process inherits the same privileges that the user has. And, while a process can have fewer privileges than the user (i.e. go down in privilege), it is rare (and generally undesirable) for a user to escalate in privilege. Information included in the privilege change graph can be determined by examining the parent of each running process, and determining whether there is a match in privilege between the parent and the child. If the privileges are different, a privilege change has occurred (whether a change up or a change down). The application ntpd is one, rare, example of a scenario in which a process escalates (1502) to root, and then returns back (1504). The sudo command is another example (e.g., used by an administrator to temporarily have a higher privilege). As with the other examples, ntpd’s privilege change actions, and the legitimate actions of various administrators (e.g., using sudo) will be incorporated in to a baseline model by platform 102. When deviations occur, such as where a new application that is not ntpd escalates privilege, or where an individual that has not previously/does not routinely use sudo does so, such behaviors can be identified as anomalous.

**[00100]** Figure 16 illustrates an example of a portion of a user login graph, which identifies which users log into which logical nodes. Physical nodes are clustered into a logical

machine cluster, for example, using yet another graph, a machine-server graph, an example of which is shown in Figure 17. For each machine, a determination is made as to what type of machine it is, based on what kind(s) of workflows it runs. As one example, suppose a given machine runs as a master node (it has a typical set of workflows it runs, as a master node). Worker nodes are different from the master node, for example, because they run Docker containers.

## **[00101] B. Polygraph Construction**

**[00102]** As previously mentioned, the polygraph depicted in Figure 7 corresponds to activities in a data center in which, in a given hour, approximately 500 virtual machines collectively run one million processes, and make 100 million connections in that hour. The polygraph represents a drastic reduction in size (e.g., from tracking information on 100 million connections in an hour, to a few hundred nodes and a few hundred edges). Using techniques described herein, such a polygraph can be constructed (e.g., using commercially available computing infrastructure) within a few minutes. The following section describes various example infrastructure that can be used in polygraph construction, and also describes various techniques that can be used to construct polygraphs.

### **[00103] 1. Example Infrastructure**

**[00104]** Returning to Figure 1, embodiments of platform 102 are built using AWS infrastructure as a service (IaaS). For example, platform 102 can use Simple Storage Service (S3) for data storage, Key Management Service (KMS) for managing secrets, Simple Queue Service (SQS) for managing messaging between applications, Simple Email Service (SES) for sending emails, and Route 53 for managing DNS. Other infrastructure tools can also be used. Examples include: orchestration tools (e.g., Kubernetes or Mesos/Marathon), service discovery tools (e.g., Mesos-DNS), service load balancing tools (e.g., marathon-LB), container tools (e.g., Docker or rkt), log/metric tools (e.g., collectd, fluentd, kibana, etc.), big data processing systems (e.g., Spark, Hadoop, AWS Redshift, Snowflake etc.), and distributed key value stores (e.g., Apache Zookeeper or etcd2).

**[00105]** As previously mentioned, in various embodiments, platform 102 makes use of a collection of microservices. Each microservice can have multiple instances, and is configured to recover from failure, scale, and distribute work amongst various such instances, as applicable. For example, microservices are auto-balancing for new instances, and can distribute workload if new instances are started or existing instances are terminated. In various embodiments, microservices are deployed as self-contained Docker containers. A Mesos-Marathon or Spark framework can be used to deploy the microservices (e.g., with Marathon monitoring and restarting failed instances of microservices as needed). The service, etcd2 can be used by microservice instances to discover how many peer instances are running, and used for calculating a hash-based scheme for workload distribution. Microservices are configured to publish various health/status metrics to either an SQS queue, or etcd2, as applicable. Amazon DynamoDB can be used for state management. Additional information on various microservices used in embodiments of platform 102 is provided below.

**[00106] a. GraphGen**

**[00107]** GraphGen (146) is a microservice that is responsible for generating raw behavior graphs on a per customer basis periodically (e.g., once an hour). In particular, Graphgen generates graphs of entities (as the nodes in the graph) and activities between entities (as the edges).

**[00108]** GraphGen performs joins on data collected by the agents, so that both sides of a behavior are linked. For example, suppose a first process on a first virtual machine communicates with a second process on a second virtual machine. Respective agents on the first and second virtual machines will each report information on their view of the communication (e.g., the PID of their respective processes). When GraphGen performs a join on the data provided by both agents, the graph will include a node for each of the processes, and an edge indicating communication between them (as well as other information, such as the directionality of the communication – i.e. which process acted as the server and which as the client in the communication).

**[00109]** Graphs created by GraphGen are written to database 142 and cached for further processing. The graph is a summary of all activity that happened in the time interval. As each

graph corresponds to a distinct period of time, different rows can be aggregated to find summary information over a larger timestamp. And, picking two different graphs from two different timestamps can be used to compare different periods. If necessary, GraphGen can parallelize its workload (e.g., if its backlog cannot be handled within an hour, or if is required to process a graph spanning a long time period).

**[00110]** GraphGen can be implemented in any appropriate programming language, such as Java or C, and machine learning libraries, such as Spark's MLLib. Example ways that GraphGen computations can be implemented include using SQL or Map-R, using Spark or Hadoop.

**[00111] b. sshTracker**

**[00112]** sshTracker (148) is a microservice that is responsible for following ssh connections and process parent hierarchies to determine trails of user ssh activity. Identified ssh trails are placed by the sshTracker into database 142 and cached for further processing.

**[00113]** sshTracker can be implemented in any appropriate programming language, such as Java or C, and machine libraries, such as Spark's MLLib. Example ways that sshTracker computations can be implemented include using SQL or Map-R, using Spark or Hadoop.

**[00114] c. ThreatAggr**

**[00115]** ThreatAggr (150) is a microservice that is responsible for obtaining third party threat information from various applicable sources, and making it available to other microservices. Examples of such information include reverse DNS information, GeoIP information, lists of known bad domains/IP addresses, lists of known bad files etc. As applicable, the threat information is normalized before insertion into database 142. ThreatAggr can be implemented in any appropriate programming language, such as Java or C, using SQL/JDBC libraries to interact with database 142 (e.g., for insertions and queries).

**[00116] d. Hawkeye**

**[00117]** Hawkeye (152) is a microservice that acts as a scheduler and can run arbitrary jobs organized as a directed graph. Hawkeye ensures that all jobs for all customers are able to

run every hour. It handles errors and retrying for failed jobs, tracks dependencies, manages appropriate resource levels, and scales jobs as needed. Hawkeye can be implemented in any appropriate programming language, such as Java or C. A variety of components can also be used, such as open source scheduler frameworks (e.g., Airflow), or AWS services (e.g., the AWS Data pipeline) which can be used for managing schedules.

**[00118] e. GBM**

**[00119]** Graph Behavior Model (GBM) (154) is a microservice that computes Polygraphs. The GBM takes as input a raw graph (e.g., as generated by GraphGen 146), nodes are actors of a behavior, and edges are the behavior relationship itself. For example, in the case of communication, example actors include processes, which communicate with other processes. GBM clusters the raw graph based on behaviors of actors and produces a summary (the Polygraph). The Polygraph summarizes behavior at a data center level. GBM also produces “observations” that represent changes detected in the data center. Such observations are based on differences in cumulative behavior (e.g., the baseline) of the data center with its current behavior. GBM can be implemented in any appropriate programming language, such as Java, C, or Golang, using appropriate libraries (as applicable) to handle distributed graph computations (handling large amounts of data analysis in a short amount of time). Spark’s MapR is another example way to compute Polygraphs.

**[00120] f. GBM Runner**

**[00121]** GBM Runner (156) is a microservice that is responsible for interfacing with GBM 154 and providing GBM 154 with raw graphs (e.g., using SQL to push any computations it can to database 142). GBM Runner 156 also inserts Polygraph output from GBM 154 to database 142. GBM Runner can be implemented in any appropriate programming language, such as Java or C, using SQL/JDBC libraries to interact with database 142 to insert and query data.

**[00122] g. EventGen**

**[00123]** EventGen (158) is a microservice that is responsible for generating alerts. It examines observations (e.g., produced by GBM 154) in aggregate, deduplicates them, and scores them. Alerts are generated for observations with a score exceeding a threshold. EventGen 158

also computes (or retrieves, as applicable) data that a customer (e.g., Alice or Bob) might need when reviewing the alert. EventGen can be implemented in any appropriate programming language, such as Java or C, using SQL/JDBC libraries to interact with database 142 to insert and query data. In various embodiments, EventGen also uses one or more machine learning libraries, such as Spark's MLLib (e.g., to compute scoring of various observations).

**[00124] h. QsJobServer**

**[00125]** QsJobServer (160) is a microservice that looks at all the data produced by platform 102 for an hour, and compiles a materialized view (MV) out of the data to make queries faster. The MV helps make sure that the queries customers most frequently run, and data that they search for, can be easily queried and answered. QsJobServer 160 also precomputes and caches a variety of different metrics so that they can quickly be provided as answers at query time. QsJobServer 160 can be implemented using any appropriate programming language, such as Java or C, using SQL/JDBC libraries. The QsJobServer is able to compute an MV efficiently at scale, where there could be a large number of joins. An SQL engine, such as Oracle can be used to efficiently execute the SQL, as applicable.

**[00126] i. Alert Notifier**

**[00127]** Alert notifier 162 is a microservice that takes alerts produced by EventGen 158 and sends the alerts to customers' integrated SIEM products (e.g., Splunk, Slack etc.). Alert notifier 162 can be implemented using any appropriate programming language, such as Java or C. Alert notifier 162 can be configured to use an email service (e.g., AWS SES or pagerduty) to send emails. Alert notifier 162 also provides templating support (e.g., Velocity or Moustache) to manage templates and structured notifications to SIEM products.

**[00128] j. Reporting Module**

**[00129]** Reporting module 164 is a microservice responsible for creating reports out of customer data (e.g., daily summaries of events, etc.) and providing those reports to customers (e.g., via email). Reporting module 164 can be implemented using any appropriate programming language, such as Java or C. Reporting module 164 can be configured to use an email service (e.g., AWS SES or pagerduty) to send emails. Reporting module 164 also provides templating

support (e.g., Velocity or Moustache) to manage templates (e.g., for constructing HTML-based email).

**[00130] k. Web Frontend**

**[00131]** Web app 120 is a microservice that provides a user interface to data collected and processed on platform 102. Web app 120 provides login, authentication, query, data visualization, etc. features. Web app 120 includes both client and server elements. Example ways the server elements can be implemented are using Java DropWizard or Node.js to serve business logic, and a combination of JSON/HTTP to manage the service. Example ways the client elements can be implemented are using frameworks such as React, Angular, or Backbone. JSON, jQuery, and JavaScript libraries (e.g., underscore) can also be used.

**[00132] l. Query Service**

**[00133]** Query service 166 is a microservice that manages all database access for web app 120. Query service 166 abstracts out data obtained from database 142 and provides a JSON based REST API service to web app 120. Query service 166 generates SQL queries for the REST APIs that it receives at run time. Query service 166 can be implemented using in any appropriate programming language, such as Java or C and SQL/JDBC libraries, or an SQL framework such as jOOQ. Query service 166 can internally make use of a variety of types of databases, including postgres, AWS Aurora (168) or Snowflake (142) to manage data for clients. Examples of tables that query service 166 manages are OLTP tables and data warehousing tables.

**[00134] m. Redis**

**[00135]** Redis (170) is an open source project which provides a key-value store. Platform 102 can use Redis as a cache to keep information for frontend services about users. Examples of such information include valid tokens for a customer, valid cookies of customers, the last time a customer tried to login, etc.

**[00136] 2. Example Processing**

# GBM

## 1. Overview

### 1.1. Concept

The GBM finds clusters of nodes in a graph that are considered to be similar based on some set of their properties and relationships to other nodes. These clusters and their relationships are used to provide visibility into the data center environment without requiring any user specified labels. GBM tracks such clusters over time persistently in order to detect changes, which can be used for alerting.

GBM operates in a batch matter where it receives the nodes and edges of a graph for a particular time period along with its previous state as input and in return it generates cluster nodes, cluster membership edges, cluster-to-cluster edges, events and its next state as output.

GBM need not try to consider all types of entities and their relationships that may be available in the conceptual “common graph” all at once. Instead, GBM has a concept of models where a subset of node and edge types and their properties are considered in a given model. One reason for this approach is scalability. Another is that trying to cluster entities in a more complex and larger graph could result in less useful results as entities may not be considered similar when looking at all possible properties and relationships they may have.

GBM can be used with different models corresponding to different subgraphs, and the core abstractions are the same:

- Each node type in a GBM model is considered to belong to a class
  - The class could be thought to represent a higher/different level of clustering where all nodes with the same class could initially considered to belong to and GBM may split them up based on the criteria it uses for this model.
  - The class for a node is represented as a string whose value is derived from the node's key and properties depending on the GBM Model. Note that different GBM models may create different class values for the same node.
- For each node type in a GBM model, GBM may generate clusters of nodes for that type
- In some embodiments, a GBM generated cluster for a given member node type cannot span more than one class for that node type.
- GBM generates edges between clusters that have the same types as the edges between source and destination cluster node types.

The approaches used are the same across models as well though different models can be configured with different settings (including settings specific to an approach or which approach to use).

The node types and the edge types may correspond to existing types in the common graph node and edge tables but this is not necessary. Even when there is a correspondence, the properties provided to GBM are not limited to the properties that are stored in the corresponding graph table entries. They can be enriched with additional information before being passed to GBM.

## 1.2. Graph Input

Logically, the input for a GBM model is characterized as follows:

- Edge triplet types (i.e. array of (source node type, edge\_type, destination node type))
- Node properties for each node type
- Edge properties for each edge type

This does not mean that the “physical” input to the GBM conforms a generic format independent of the model. For instance, the edges in the PtypeConn model correspond to edges between the MN clusters (each process node has an MN cluster id property). In the Uid2Uid2 model, edges are not explicitly provided separately from nodes (the euid array in the node properties serve the same purpose). However, in both case, the physical information provides the “logical” input mentioned above.

An example GBM implementation supports models that have a single edge triplet type but the abstraction is not limited to that and the design can be extended as applicable.

## 1.3. State Input

The state input for GBM for a particular model is stored in a file. Except for when GBM runs for the first time for a given model or it is reset, a state file (from its previous run) is provided to it along with the graph data.

Sometimes there is no data available for a particular model in a given time period and GBM may not be run for that time period. When data is available in a future time period, GBM will be run with the latest state file provided to it as input.

## 1.4. Graph Output

GBM outputs cluster nodes, cluster membership edges and inter-cluster relationship edges that are stored in the following graph tables node\_c, node\_cm and node\_icr, respectively.

The type names of nodes and edges obey the following rules:

- A given node type may be used in two different GBM models. The type names of the cluster nodes generated by two such models for that node type will be different. For

instance, Process type nodes appear in both PtypeConn and Uid2Uid model but their cluster nodes will have different type names.

- The membership edge type name is "MemberOf".
- The edge type names for cluster-to-cluster edges will be the same as the edge type names in the underlying node-to-node edges in the input.

### 1.5. Event Types

GBM generates the following types of events:

- New class
- New cluster
- New edge from class to class
- Split class (the notion is that GBM considers all nodes of a given type and class to be in the same cluster initially and if it splits them into different clusters, it is splitting a class)
- New edge from cluster and class
- New edge between cluster and cluster
- New edge from class to cluster

One underlying node or edge in the logical input could cause multiple types of events to be generated; conversely, one event could correspond to multiple nodes or edges in the input.

Not every model generates every event type.

## 2. GBM Models

### 2.1. PTypeConn Model

The node input to the this model for a given time period comprises non-interactive (i.e. not associated tty) Process nodes that had connections in that period and the base graph nodes of other types (IpSep, DnsSep and IpAddress) that have been involved in those connections. In some embodiments, gbm-runner does not provide MachineUc to gbm as it is not included in the MN process cluster edges. In some embodiments, MachineSep is provided to gbm but it is not used by gbm for clustering.

The base relationship is the connectivity relationship for the following type triplets:

- Process, ConnectedTo, Process
- Process, ConnectedTo, IpSep
- Process, ConnectedTo, DnsSep
- IpAddress, ConnectedTo, ProcessProcess

The edge input to this model is the MN process cluster ConnectedTo edges instead of individual node-to-node ConnectedTo edges from base graph. This works along with providing the mn cluster id for each node and saves IO between gbm-runner and gbm.

The membership edges created by this model refer to the base graph node type provided in the input.

This model clusters nodes of the same class that have similar connectivity relationships. For example, if two processes had similar incoming neighbors of the same class and outgoing neighbors of the same class, they could be clustered.

#### 2.1.1. Class Values

The class values of nodes are determined as follows depending on the node type:

##### 2.1.1.1. Process nodes

- if exe\_path contains java then "java <cmdline\_term\_1> ..."
- else if exe\_path contains python then "python <cmdline\_term\_1> ..."
- else "last\_part\_of\_exe\_path"

##### 2.1.1.2. IpSep nodes

- if ip\_internal then "IntIpS"
- else if severity = 0 then "<ip\_addr>:<protocol>:<port>"
- else "<ip\_addr>:<port>\_BadIp"

##### 2.1.1.3. DnsSep nodes

- if ip\_internal = 1 then "<hostname>"
- else if severity = 0 then "<hostname>:<protocol>:port"
- else "<hostname>:<port>\_BadIp"

##### 2.1.1.4. IpAddress nodes (will appear only on client side)

- if ip\_internal = 1 then
  - "IpIntC"
- else if severity = 0 then "ExtIpC"
- else "ExtBadIpC"

#### 2.1.2. Events

A new class event in this model for a Process node is equivalent to seeing a new ctype being involved in a connection for the first time. Note that this does not mean the ctype was not seen before. It could have if it did not make a connection before.

A new class event in this model with for an IpSep node with ip\_internal = 0 is equivalent to seeing an connection to a new external IP address for the first time.

A new class event in this model for a DnsSep node is equivalent to seeing an connection to a new DNS name for the first time.

A new class event in this model for an IpAddress node with ip\_internal = 0 and severity = 0 is equivalent to seeing a connection from any external ip address for the first time.

A new class event in this model for an IpAddress node with ip\_internal = 0 and severity > 0 is equivalent to seeing a connection from any bad external ip address for the first time.

A new class to class to edge from a class for a process node to a class for a process node equivalent seeing communication the source ctype making a connection to the destination ctype for the first time.

A new class to class to edge from a class for a process node to a class for a DnsSep node equivalent seeing communication the source ctype making a connection to the destination Dns name for the first time.

## 2.2. IntPConn Model

This model is similar to PtypeConnModel except that connection edges between parent/child processes and connections between processes where both sides are not interactive are filtered out.

## 2.3. Uid2Uid Model

The node input to the this model for a given time period comprises Process nodes that are running in that period.

The value of class of Process nodes is "<username>".

The base relationship that is used for clustering is privilege change either by the process changing its effective user id or by launching a child process which runs with a different user.

This model clusters processes with the same username and shows similar privilege change behavior. For instance, if two processes with the same username had similar effective user values, launched processes with similar usernames and were launched by processes with similar username, then they could be clustered.

The physical input for this model comprises Process nodes only with the caveat that the complete ancestor hierarchy of Process nodes active (i.e. running) for a given time period is provided as input even if an ancestor is not active in that time period. Note that effective user ids of a process are represented as an array in the Process node properties and launch relationships are available from ppid\_hash fields in the properties as well.

An edge between a source cluster and destination cluster generated by this model means that all of the processes in the source cluster had a privilege change relationship to at least one process in the destination cluster.

### 2.3.2. Events

A new class event in this model is equivalent to seeing a user for the first time.

A new class to class edge event is equivalent seeing the source user making a privilege change to the destination user for the first time.

## 2.4. Ct2Ct Model

The node input to the this model for a given time period comprises Process nodes that are running in that period.

The value class of Process nodes is ctype (similar to how it is created for PtypeConn model).

The base relationship that is used for clustering is a parent process with a given ctype launching a child process with another given destination ctype.

This model clusters processes with the same ctype and show similar launch behavior. For instance, if two processes with the same ctype had launched processes with similar ctypes, then they could be clustered.

The physical input for this model comprises Process nodes only with the caveat that the complete ancestor hierarchy of Process nodes active (i.e. running) for a given time period is provided as input even if an ancestor is not active in that time period. Note that launch relationships are available from ppid\_hash fields in the Process node properties.

An edge between a source cluster and destination cluster generated by this model means that all of the processes in the source cluster launched at least one process in the destination cluster.

### 2.4.2. Events

A new class event in this model is equivalent to seeing a ctype for the first time. Note that the same type of event will be generated by the PtypeConn model as well.

A new class to class edge event is equivalent seeing the source ctype launching the destination ctype for the first time.

## 2.5. MtypeConn Model

The node input to the this model for a given time period comprises Machine Nodes that had connections in that period and the base graph nodes of other types (IpAddress and DnsName) that have been involved in those connections.

The base relationship is the connectivity relationship for the following type triplets:

- Machine, ConnectedTo, Machine
- Machine, ConnectedTo, IpAddress
- Machine, ConnectedTo, DnsName
- IpAddress, ConnectedTo, Machine

The edge input to this model are the corresponding ConnectedTo edges in the base graph.

The membership edges generated by this model will refer to Machine, IpAddress, DnsName and IpSep nodes in the base graph. Though the nodes provided to this model are IpAddress nodes instead of IpSep nodes, the membership edges it generates will refer to IpSep type nodes. Alternatively, the base graph can generate edges between Machine and IpSep node types. Note that the Machine to IpAddress edges have tcp\_dst\_ports/udp\_dst\_ports properties that can be used for this purpose.

#### 2.5.1. Class Values

##### 2.5.1.1. Machine

The class value for all Machine nodes is "Machine".

In some embodiments, the machine\_terms property in the Machine nodes is not used for clustering but is optionally used for labeling machines that are clustered together. If a majority of machines clustered together share a term in the machine\_terms, that term can be used for labeling the cluster.

##### 2.5.1.2. IpSep

The class value for IpSep nodes can be determined as follows:

```
if ip_internal then "IntIpS"
else
    if severity = 0 then "<ip_addr>:<protocol>:<port>"
    else "<ip_addr_BadIp>"
```

##### 2.4.1.3. IpAddress

The class value for IpAddress nodes can be determined as follows:

```
if ip_internal then "IntIpC"
else
    if severity = 0 then "ExtIpC"
    else "ExtBadIpC"
```

##### 2.4.1.3. DnsName

The class value for DnsName nodes can be determined as follows:

```
if severity = 0 then "<hostname>"
else then "<hostname>_BadIp"
```

This model clusters nodes of the same class that have similar connectivity relationships. For example, if two machines had similar incoming neighbors of the same class and outgoing neighbors of the same class, they could be clustered.

A new class event in this model will be generated for external IP addresses or Dns names seen for the first time. Note that there a new class to class to edge Machine class to a class for an IpSep or DnsName node will also be generated at the same time.

## 3. GBM Event Types

### 3.1. New Class Event

#### 3.1.1. Structure

The key field for this event type looks as follows using PtypeConn model as an example:

```
{
  "node": {
    "class": {
      "cid": "httpd"
    },
    "key": {
      "cid": "29654"
    },
    "type": "PtypeConn"
  }
}
```

It contains not only the class value but also the id of the cluster where that class value is observed. Since multiple clusters can be observed with the same value in a given time period, GBM can generate multiple events of this type for the same class value.

The props field looks as follows:

```
{
  "set_size": 5
}
```

The set\_size indicates the size of the cluster referenced in the keys field.

#### 3.1.2. Conditions

For a given model and time period, multiple NewClass events can be generated if there is more than one cluster in that class.

NewNode events will not be generated separately in this case.

### 3.2. New Class to Class Edge Event

#### 3.2.1. Structure

The keys field for this event type looks as follows using PtypeConn model as an example:

```
"edge": {
  "dst_node": {
    "class": {
      "cid": "java war"
    },
  },
}
```

```

    "key": {
      "cid": "27635"
    },
    "type": "PtypeConn"
  },
  "src_node": {
    "class": {
      "cid": "IntIpC"
    },
    "key": {
      "cid": "20881"
    },
    "type": "PtypeConn"
  },
  "type": "ConnectedTo"
}

```

The key field contains not only the src and dst class values but also the src and dst cluster ids (i.e. the src/dst\_node:key.cid represents the src/dst cluster id)

In a given time period for a given model, an event of this type could involve multiple edges between different cluster pairs that have the same src/dst class values. GBM generates multiple events in this case with different src and dst cluster ids.

The props fields looks as follows for this event type:

```

{
  "dst_set_size": 2,
  "src_set_size": 1
}

```

The src/dst\_set\_size represents the sizes of the clusters given in the keys field.

### 3.2.2. Conditions

For a given model and time period, multiple NewClassToClass events can be generated if there is more than one pair of clusters in that class pair.

NewNodeToNode events will not be generated separately in this case.

## 3.3 Combining Events At Class Level

For a given model and time period, the following types of events can represent multiple changes in the underlying gbm cluster level graph in terms of multiple new clusters or multiple new edges between clusters

- NewClass
- NewEdgeClassToClass
- NewEdgeNodeToClass
- NewEdgeClassToNode

Multiple NewClass events with the same model and class can be output if there are multiple clusters in that new class.

Multiple NewEdgeClassToClass events with the same model and class pair can be output if there are multiple new cluster edges within that class pair.

Multiple NewEdgeNodeToClass events with the same model and destination class can be output if there are multiple new edges from the source cluster to the destination clusters in that destination class (first time seeing this class as a destination cluster class for the source cluster).

Multiple NewEdgeClassToNode events with the same model and source class can be output if there are multiple new edges from source clusters in that to the destination clusters in that source class (first time seeing this class as a source cluster class for the destination cluster).

These events may be combined at class level and treated as a single event when it is desirable to view changes at the class level, e.g. when one wants to know when there is a new ctype.

Also note that different models may have partial overlap in the types of nodes they use from from the base graph. Therefore, they can generate NewClass type of events for the same class. Thus NewClass events can also be combined across models when it is desirable to view changes at the class level

## 4. Additional Information

### Connection Classification

Connection classification classifies following connections into connection type.

Connection type. Connection type can be one of (Interactive, non-interactive)

The classifier is trained using in-house data like inter-packet delay, histogram of bytes transferred, sessions etc. to create a supervised, ML model. The model is able to predict whether a session is interactive or not.

The output of the classifier is pushed to Snowflake and will be available for queries.

### User identity classification

User identity does not typically have a uniform view across all machines. Assume that MID:UID uniquely identifies a user on that machine. The name of the user may change, but UID does not get reused for different users. User classification produces the following classification.

User Id. A Global user identity that can be compared across machines. This should map to a set of [MID:UID] pairs.

Input to the classifier will be user, group information and, in some embodiments, ADC information of different users.

#### Application Labeling

Application labeling takes the following information into account and predict the Application label of every process in the system.

1. Cmdline
2. Pkg information
3. List of listen ports.

Label predicted, can be used to pre-populate default label for application for the process.

#### Entity Classification

Classification service creates the canonical graph. The service classifies incoming data periodically, into the following types

1. User Type
2. Process type
3. Container Type
4. Machine Type

The output of classification is maintained as a log in snowflake. i.e. a particular pid can change process type over time and when doing queries, latest type has to be obtained.

Since classification happens periodically, a change in classification can signal a change in behavior of the process. Classification may find new process, machine or apps and it may not have enough to classify the entity. In this case entities will remain unclassified-pending new data.

Entity classifier can be further divided into steps. First classifier produces a 'Signature' per entity. This signature can be compared across time intervals and can be compared with other signatures using a 'distance function' to get a similarity probability. A graph-clustering approach can then cluster various entities into 'Similar' buckets based on the distance function. Since signatures can be compared across time periods, this means clustering can also be compared across different time periods.

#### Signature table

|                                   | Input                                                                                                                                                                                                               | Output                                                             |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------|
| Process signature                 | For every process: <ul style="list-style-type: none"><li>• Cmdline</li><li>• PGID [process record]</li><li>• PPID [process record]</li><li>• PKGINFO</li><li>• Descendants</li><li>• List of LISTEN ports</li></ul> | A table of [PID, Signature]<br>A function (Sig1, Sig2) -> distance |
| Entity network behavior signature | For every entity: <ul style="list-style-type: none"><li>• Node connection graph.</li><li>• Listen ports.</li></ul>                                                                                                  | A table of [PID, Signature]<br>A function (Sig1, Sig2) -> distance |

---

- Connection\_type

|                        |                                                                                       |                  |
|------------------------|---------------------------------------------------------------------------------------|------------------|
| Process Classification | Process signature table<br>Process network behavior signature<br>Application labeling | Ptype<br>AppType |
| Machine Classification | Ptypes on the machine<br>Machine network behavior<br>signature                        | MType            |
| User Classification    | User network behavior signature<br>Ptypes                                             | UserType         |

---

In case of User-Type, 3 buckets are used.

1. Interactive
2. Non-interactive
3. Mixed use.

## PType

- Features
  - Able to detect behavior changes across data center
  - Able to understand and reason about how the behavior has changed over time
  - Measure the change to see if something is expected or not
- Ptype is a clustering of known processes in a datacenter and describes their network relationships as a graph
  - Similarly behaving processes are grouped together
- Behavior of Ptypes can change over time
  - Group of processes can change behavior over time
  - System should learn behavior changes over time
- I.e. Ptype **'does not'** describe a behavior that remains static over time

## CType

- CType is a clustering of known commands in a datacenter
  - Command Lines with similar Process properties are grouped together
- Random Forest based clustering
  - Command line of the process (Uses TFID internally)
  - Uptime (> 1 min or less)
  - Any children started
  - UID, EUID and change if any
  - TTY, change if any
  - Listen ports < 32,000 or count of listen ports
- Split
  - EXE\_PATH for CType must be same for all members
- Predictor model
  - Predicts the CType of any new process or declares it unknown
    - Matches all features and uses Levenstien distance for Cmdline
  - Generates stable CType. I.e. If process features haven't changed, CType predicted is the same
- Incrementally maintained over time
  - Every hour, re-cluster based on seeded/labeled/sampled historical data + new process data only

As much time invariant properties as possible  
Data available within first 1 min of process activity

# SimRank

- Cluster processes based on their similarity of network structure
- Computes 'Similarity' of nodes as follows
- Rules based clusters
  - Matching Neighbors
  - DNS clustering
  - Local-Connections-Only processes are eliminated and grouped under Ctype

## I-Sim(A,B) and O-Sim(A,B)

- 'Nodes with similar neighborhood graphs are similar'
- Averaged over the whole graph iteratively

## Input-Output(A, B)

- $X = I-Sim(A,B)$
- $Y = O-Sim(A,B)$

## Affinity propagation

## Split

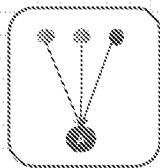
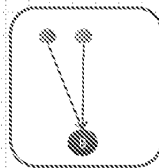
- Ctype Split
- Exact network split

$I-Sim(A,B) = \text{HungarianWeighted}[\text{Input}(A), \text{Input}(B)]$   
 $I-Sim(A,A) = 1$   
 $I-Sim(A,B) = 0$  if  $\text{Input}(A)$  or  $\text{Input}(B)$  are nil  
 $I-Sim(A,B) = \text{Nil}$  if  $\text{Input}(A)$  and  $\text{Input}(B)$  are nil

If  $X$  is nil and  $Y$  is nil { return  $X + 0.12$  }  
 else return  $X$  or  $Y$  whichever is not nil

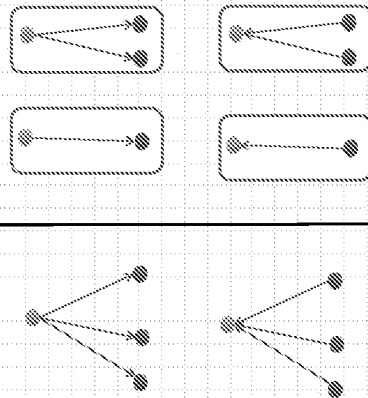
Every member of five  
 set should have connected to 1  
 member of set 2

# Hungarian weighted similarity



- Comparing 2 nodes (Blue nodes)
- $Sim(\text{Orange}) = 0.6$
- $Sim(\text{Green}) = 0.4$
- $Sim(A,B) = (0.6 + 0.4 + 0)/3 = 0.33$

## Local connections only



- Orange :
  - Nodes with local connections only
- Group them by CType + Direction
  - Clients together
  - Servers together

## Cumulative Graph

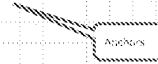
- Cumulative Graph maintains running model of PType behavior across time
  - Helps us recognize how processes behave over time
- It is bootstrapped from first hour of network similarity and then maintained
- Nodes (are a PType)
  - Maintains list of all active processes and pids in last hour
  - Maintains number of historic total processes
  - Average active processes per hour
  - Current CType of the PType
  - Historic CType and frequency
- Edges
  - Connectivity edge
    - Connectivity frequency
    - Weight of the edge (Number of connections, bytes etc)
  - Transition edge
    - Transition frequency

For now we assume CType doesn't change

User can mark these acceptable

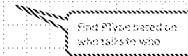
## Merge (Cumulative, Per-hour)

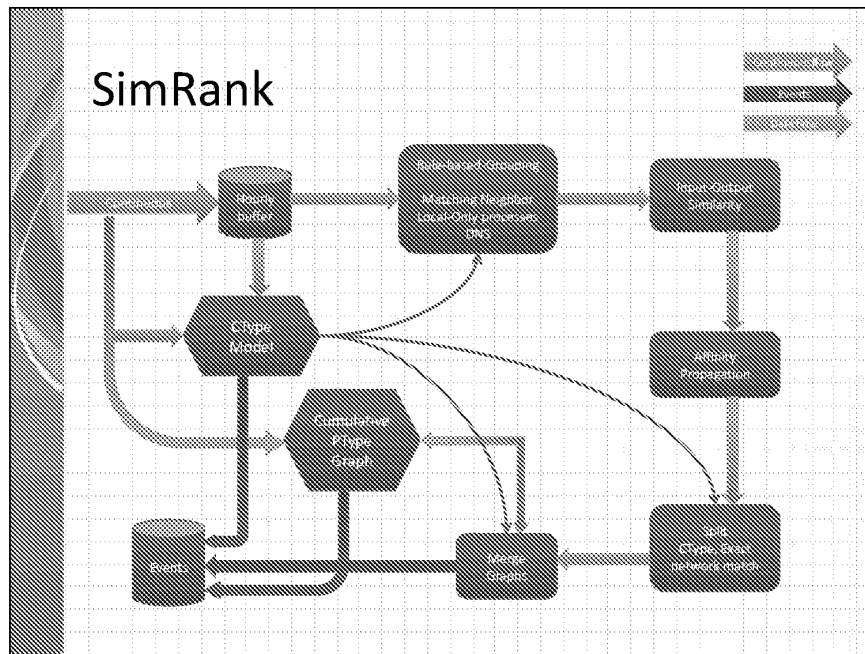
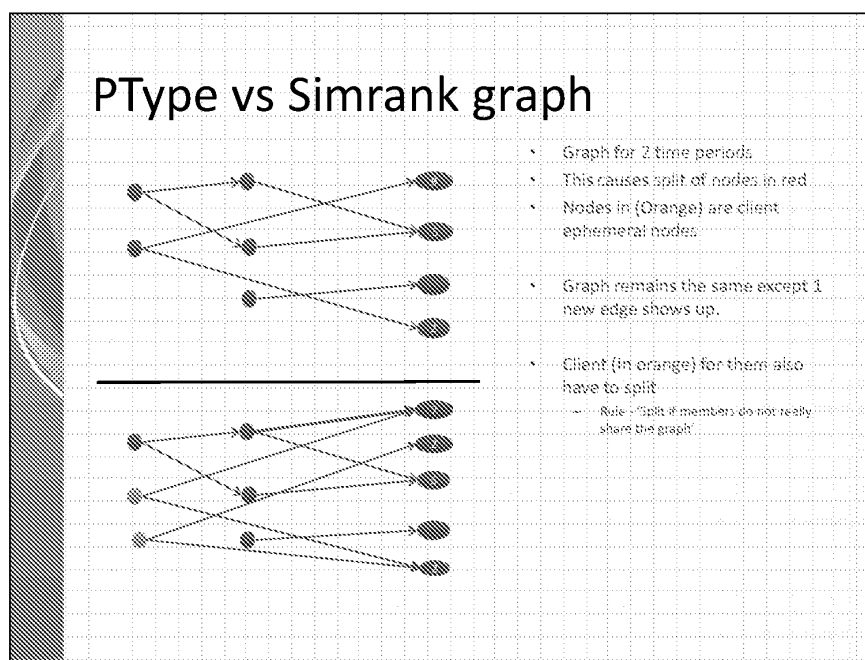
- Sort graphs on PType size(active member-list)
- For all nodes in, cumulative graph with active processes, compare with PerHour PType
  - Per-Hour PType with maximal ( $> 2/3$ ) active processes == PType as cumulative graph
  - Any new members who join the PType are deemed as same
    - If they moved from another Ptype, this is noted as a transition
  - Any members who are no longer part of set, are a different PType now
    - Again a transition



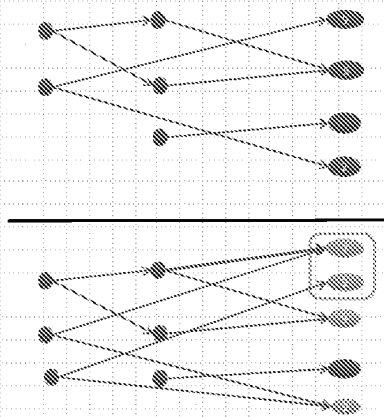
## Merge (Cumulative, Per-hour)

- While (! Convergence) {
- For (All nodes in per-hour not yet identified), compare with Cumulative graph {
  - A PType with same network signature as current one for 'Identified nodes so far'
  - For any unidentified nodes, CType and structure must match
  - CType must be in historical Ctype list
  - If any surviving members are found, it increases the strength, match the maximal set
- }}
- For any surviving node, if PType changes, note as a transition
- Any remaining PType are new PType, and can be added as new to Cumulative graph



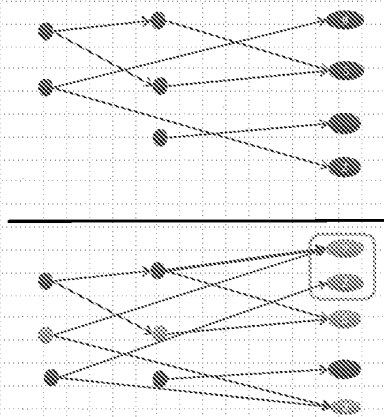


## Membership



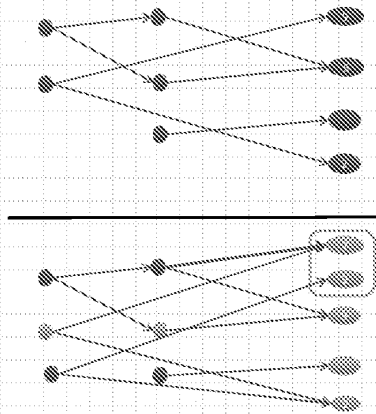
- Membership based rule:
  - Ptype with 7 members in Ptype with 9 members.
  - Change in behavior (1 new edge)
  - Ptype with 2 nodes is new ptype.
  - 2 groups still share CndType.
- Other nodes in green could be matched on membership too

## Network signature



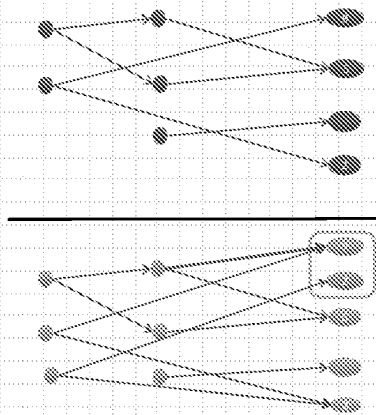
- Network signature rule:
  - Network connections to already identified (green nodes) must match
  - Connections from 'Unidentified' nodes must match 'CndType' level
  - CndType must be same
  - Weightage from 'Common members', if available
- Iterate till convergence
- Servers first, followed by clients

## Network signature



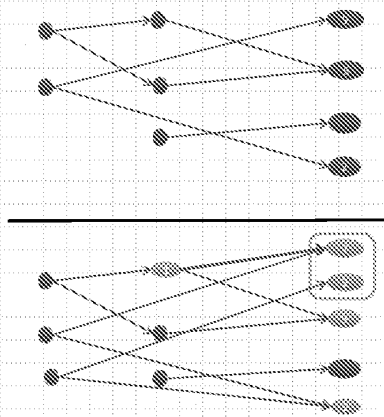
- Network signature rule:
  - Network connections to already identified (green nodes) must match
  - Connections from 'Unidentified' nodes must match 'CndType' level
  - CndType must be same
  - Weightage from 'Common members', if available
- Iterate till convergence
- Servers first, followed by clients

## Network signature



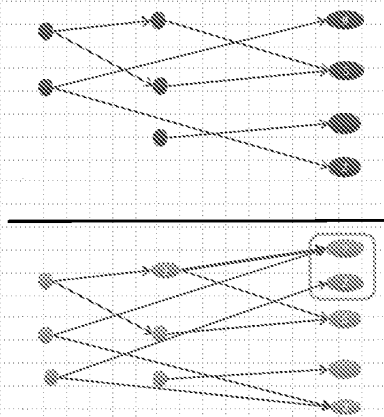
- Network signature rule:
  - Network connections to already identified (green nodes) must match
  - Connections from 'Unidentified' nodes must match 'CndType'
  - CndType must be same
  - Weightage from 'Common members', if available
- Iterate till convergence
- Servers first, followed by clients

## Membership



- If (1) had matched because of long running processes

## Network signature



- Network signature rule:
  - Network connections to already identified (green nodes) must match
  - Connections from 'Unidentified' nodes must match 'EndType'
  - EndType must be same
  - Weightage from 'Common members', if available
- Iterate till convergence
- Servers first, followed by clients

## Events - CType

- New Command (CType) in DC
- Existing CType changed
  - EUID (Priv change)
  - CType started a child
  - CType opened a Listen port
  - CType running longer than usual
  - Size/MTime of the EXE\_PATH changed
- New connection by a CType
- *Queries*
  - New CType on an existing machine
  - A user using CType for the first time

## Events - PType

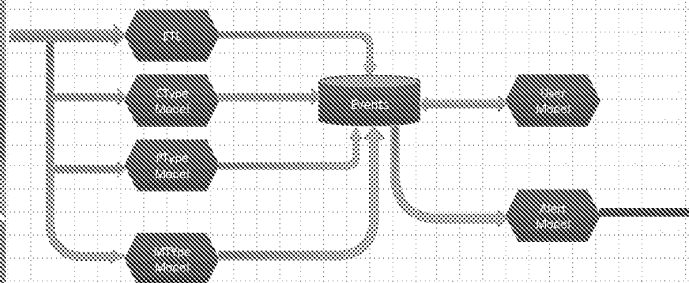
- New PType
  - But known CType
  - New CType
  - With large difference in behavior
- An existing PType behaving differently
  - Talking to Same PType
  - Talking to different PType but similar CType
  - Talking to different CType
  - Talking to a new PType but not a new CType
  - Talking to a new PType and a new CType
  - Server started a connection out
  - Client behaving like a server
  - Has a new CType
- PType transitioning to a different PType
- A member of the set behaving differently (and it never happened before)

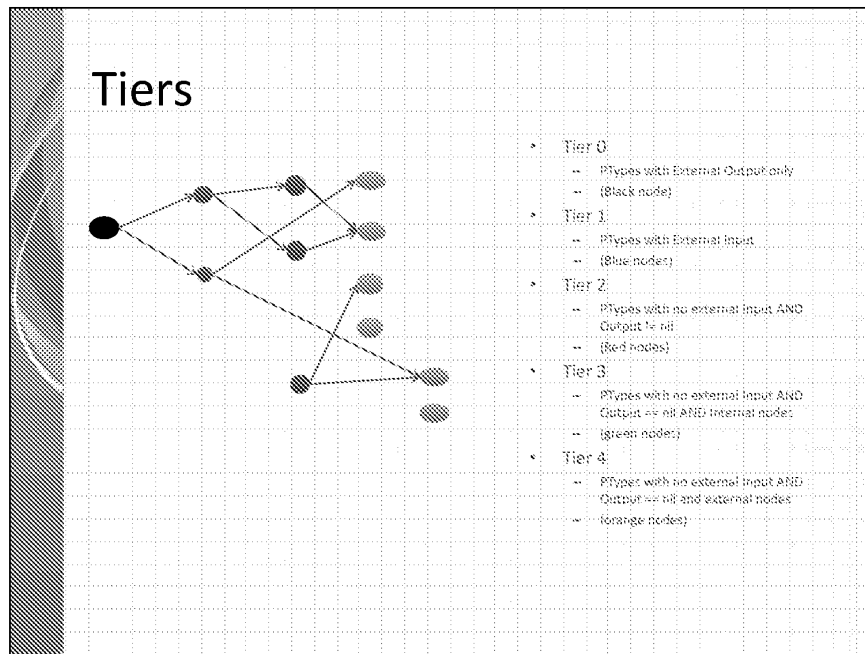
Severity = function(CType, changes differences)

## Other Information

- ETL
  - Stepping Stones
    - Connect internal nodes to get actual data destination
- Activity attributed to a Machine (MType mode)
  - DNS error activity
  - DNS server
  - Default Router
  - Promiscuous mode
  - Kernel modules loaded
  - USB insertion
  - Connections for which Process is not known
  - User Logins (Failed or successful)
- User Model
  - Create a model for users based on their activity
- Alert Model
  - Analyze the alerts for severity

## Alternate Embodiment





## Additional Polygraph Information

Create an hourly connectivity graph and clustering based on the graph. For next hour, this graph can be used as a model.

For every new process:

Predict which group the process belongs to.

For every new edge:

- a. Calculate probability of being bad.
- b. Calculate probability of process making the edge, compared to its peer.

Graph comparison:

1. Run network similarity for 2 hours, 1 each.
2. Compare servers from 1 hour to the other.
3. For each server that existed in hour1 and exists in hour2:
  - a. Calculate probability of 2 clusters being same based on their memberships.
  - b. For maximal cluster, keep clusterId the same.
  - c. For each initial cluster, one of these things could happen:
    - i. Cluster remains same 100%.
    - ii. Cluster splits into 1 main group and 1 or more smaller clusters.
    - iii. Cluster joins another cl
  - d. Calculate signature of server
  - e. Designate clusters as same

Example anomalies to detect:

- a. A new binary starts running with unique network behavior.
- b. A new binary with existing network behavior.
- c. An existing process behaves differently.
- d. A process launched from by somebody that it typically does not launch from.
- e. Amount of data transferred.
- f. New DNS query.
- g. Binary MD5 changed.
- h. Binary Size changed.

For every customer, system maintains following data structures:

1. Random Forest based model for CmdType.
2. Simrank based network similarity.

#### CmdType model

CmdType model works on process parameters. Its parameters are limited to process properties available within a short time (max 1 minutes) of process startup. It can be used to get a CmdType ID of any new process if that command has been seen before, otherwise it learns about new commands presented to it.

#### Input

CmdType model has following example input parameters about a process:

1. Command line of the process
  - a. Command line of the parent of the process (if not 0 or 1)
  - b. Command line of the SID
2. Uptime (> 1 mins or less)
3. UID, EUID and change if any
4. TTY, change if any
5. Listen ports
6. Any children

#### Approach summary

CmdType can be maintained as a Random Forest based approach. It uses TFID clustering over Cmdline to convert it into a TFID clusterid. Then also derives new information from other input parameters to create a cluster of the processes.

After classification, it can be used to create a prediction Model 'Random Tree' that predicts the cluster of a new incoming process based on input features. Cluster IDs are persisted to disk hence remains stable.

1. Feature 1- N
2. Cluster ID

#### Output

CmdType model can predict the CmdType of any new process or generate alerts.

For a given process, calculate the input features and check the on-disk prediction table. If a match is found, predict the process to be of that CmdType. Otherwise compare the CmdType with its nearest CmdType as calculated by Levenstien distance between its features. Diff the CmdType and alert the system on discovery of a new Command type with closes match and comparison.

#### *Bootstrap*

Bootstrap is as follows:

1. The first hour when customer data comes in
  - a. Calculate a full random forest CmdType prediction model.
2. Split the cluster to make sure 'exe\_path' is unique for a CmdType.

#### *Maintain/Updates*

Every hour

1. For each new process
  - a. Calculate the process features
  - b. Check to see if CmdType is known, if not known, then include in list of 'New processes'
2. Create a new RF seeded with sample of processes from prediction-table. Limit to 10-100 processes per cluster.
3. Run an 'Unsupervised' version of RF with 'Seed' + 'New processes'
4. Analyze the results, by evaluating every new process
  - a. If process is clustered into an existing set, predict that set
  - b. If a new cluster is created, add it to prediction table and generate a new CmdType ID for it

#### *Network Similarity ranking*

Similarity between 2 nodes can be defined as following formula

Input or output Similarity = Jaccard similarity Coefficient of 2 nodes

IS == IS == nil, IOS → (OS)

OS == OS == nil, IOS = → (IS)

S(A) != nil or S(B) != nil, IOS = (IS+OS)/2

IOS = IN or Output if the other set is nil, or average of 2.

#### *Split*

After sim ranking is calculated, nodes are split based on CmdType. Each individual node at this point is a 'PType' of a particular cmdtype. Every ptype is given a persistent id and stored persistently to disk as a cumulative graph.

After CmdType split, another split is done to the sets where if an edge exists between 2 PTypes, a member of 1 set must have a connection to a member of the other set with known frequency.

#### *Update*

Every hour a new network similarity graph is computed and merged with the cumulative graph as follows

1. For every cmdType compare the 2 sets in cumulative and new graph as follows

- a. For any 2 nodes that are members of CmdType,
  - i. Compare internal members and calculate set-membership Jaccard distance ( $N^2$ )
  - ii. Order the sets in decreasing order of similarity. i.e. most similar sets first, followed by second most similar etc.
  - iii. For nodes with  $> 2/3$  members, declare the new nodes as same as before (i.e. give them same physical id)
  - iv. For rest of the nodes, since there are not many members from previous period, there must be new members, or members have shifted around.
- b. After all the existing nodes that could be classified, have been classified,
- c. For all nodes in graphs not classified, create a network signature for each in terms of who they talk to
  - i. If network signature matches in 2 graphs, designates the nodes as same.
  - ii. Keep recursing, till no new nodes can be identified.
- d. Remaining nodes are new ptypes, and create a signature for them. Signature is written in terms of 'CmdType' and not ptype.
- e. For every cmdtype, if it talks
  1. Calculate 'Network signature' based on which cmdtypes
  2. If a node is new,
    - ii. Calculate Jaccard distance on their neighboring (Input and output sets)
2. After all nodes are identified, a summary of their network behavior is maintained as a probability of every edge.
3. Any ptype and its edges where all members exit is still retained in the graph. In future if a new process shows up with network signature similar to a known ptype, they can be classified.
4. After a week, if no process matches a ptype, it is dropped, unless directed by users as ok
5. A process can move from 1 ptype to a new ptype over time. The move is remembered as an edge and probability of that happening

#### Example Alerts

1. A new ptype is discovered
2. A new edge between a ptype and existing ptype.

#### Example Usage in Intrusion Detection

A new process is first categorized based on its CmdType. If a new CmdType is discovered, it is alerted. After a suitable period has passed for observing the process behavior (1 hour), process is categorized as ptype. If a new ptype is discovered, it is alerted. If a new ptype, which is also a new cmdtype, alert severity goes up. If an existing process ends up changing ptype, it is alerted. If the ptype cmdtype is different, severity of alert goes up.

# Extended User Session Tracking

## 1. Overview

One model for tracking user behavior in a distributed computing environment is the concept of an extended user session that would associate OS processes across one or machines with a single original login and thus original user. Coupled with associating actions with processes, one can associate actions with extended user sessions and track user behavior correctly even when a malicious user attempts to hide his trail by changing user identities. Described herein are techniques for the association of processes with extended user sessions.

Note that extended user session tracking may be useful in operational use cases without malicious intent if users make original logins with distinct usernames but then perform actions under a common username. A typical scenario would be where multiple users with admin privileges exist and they need to gain super-user privilege to perform a particular type of maintenance. It may be desirable to know which operations are performed by which user when debugging issues.

The methods for extended user session tracking are described below using secure shell (ssh) protocol as implemented by openssh (on the server side) as the mechanism for logins but they are not limited to ssh protocol or a particular implementation and can easily be applied to other login mechanisms.

On any given machine, there will be a process that listens for and accepts ssh connections on a given port. This process can run the openssh server program running in daemon mode or it could be running another program (e.g. initd on a Linux system). In either case, a new process running openssh will be created for every new ssh login session and this process can be used to identify a ssh session on that machine. This process is called the "privileged" process in openssh.

After authentication of the ssh session, when an ssh client requests a shell or any other program to be run under that ssh session, a new process that runs that program will be created under (i.e. as a child of) the associated privileged process. If ssh client requests port forwarding to be performed, the connections will be associated with the privileged process.

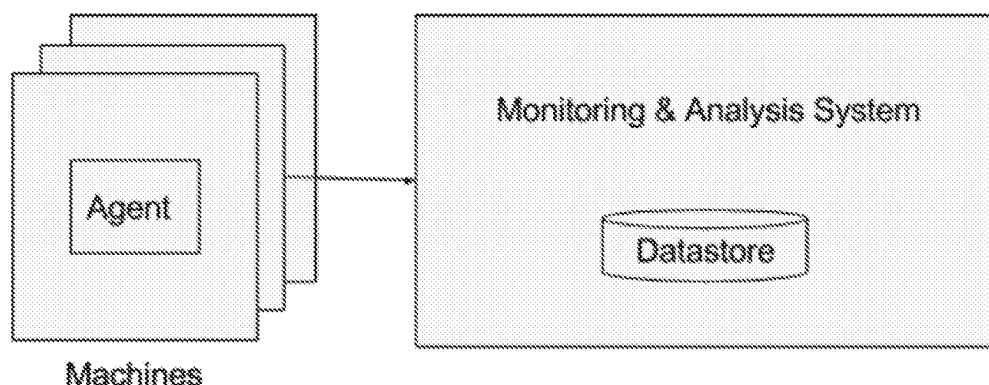
In modern operating systems such as Linux and Windows, each process has a parent process (except for the very first process) and when a new process is created the parent process is known. By tracking the parent-child hierarchy of processes, one can determine if a particular process is a descendant of a privileged openssh process and thus if it is associated with a ssh login session.

For user session tracking across machines (or a single machine with multiple logins) in a distributed environment, it is established when two login sessions have a parent-child

relationship. After that, the “original” login session, if any, for any given login session can be determined by following the parent relationship recursively.

## 2. Example Design

As shown in the high level data flow figure below, software agents running on machines detect new connections, processes and logins and send associated records to the monitoring and analysis system, which contains a datastore to persistently store input and output data.



### 2.1. Data Model

The input and output data are modeled using logical tables. These tables can be persisted in a datastore that allows querying of data, e.g. a relational database that provides a SQL interface. Other types of datastores such as graph oriented databases or hybrid schemes can be used as well.

#### 2.1.1. Common Identifiers

The following identifiers are commonly used in the tables:

- mid
- pid\_hash

The mid stands for machine id. Each machine (physical or VM) instance is assigned a number that is unique across time and space.

Operating systems use numbers called process identifiers (pid) to identify processes running at a given time. Over time processes may die and new processes may be started on a machine or the machine itself may restart. The pid is not necessarily unique across time in that the same pid value can be reused for different processes at different times. In order to track process descendants across time, one cannot simply use the pid but need to account for time as well.

In order to be able to identify a process on a machine uniquely across time, another number called pid\_hash is generated for process. The pid\_hash is generated using a collision-resistant hash function that takes the pid, start time and possibly other properties of a process.

An ssh login session can be identified uniquely by an (mid, pid\_hash) tuple.

#### 2.1.2. Input Data Model

Input data collected by the agents on the machine constitute the input data model and are represented by the following logical tables:

- connections
- processes
- logins

##### 2.1.2.1. Connections Table

The connections table maintains records of TCP/IP connections observed on each machine. It has the following columns:

| Column Name | Description                                                            |
|-------------|------------------------------------------------------------------------|
| mid         | Id of the machine that the connection was observed on                  |
| start_time  | Connection start time                                                  |
| pid_hash    | Id of the process that was associated with the connection              |
| src_ip_addr | Source IP address (connection was initiated from this IP address)      |
| src_port    | Source port                                                            |
| dst_ip_addr | Destination IP address (connection was made to this IP address)        |
| dst_port    | Destination port                                                       |
| prot        | Protocol (TCP or UDP)                                                  |
| dir         | Direction of the connection (incoming or outgoing) wrt to this machine |

The source fields always represent the side where the connection was initiated from.

On the destination side, the agent associates a ssh connection with the privileged ssh process that is created for that connection.

For each connection in the system, there would be two records in this table assuming that the machines on both side of the connection capture the connection. These records can be matched based on equality of the tuple (src\_ip\_addr, src\_port, dst\_ip\_addr, dst\_port, protocol) and proximity of the start\_time fields.

#### *2.1.2.2. Processes Table*

The processes table maintains records of processes observed on each machine. It has the following columns:

| Column Name | Description                                        |
|-------------|----------------------------------------------------|
| mid         | Id of the machine that the process was observed on |
| pid_hash    | Id of the process                                  |
| start_time  | Start time of the process                          |
| exe_path    | The executable path of the process                 |
| ppid_hash   | Id of the parent process                           |

#### *2.1.2.3. Logins Table*

The logins table maintains records of logins to the machines. It has the following columns:

| Column Name    | Description                                             |
|----------------|---------------------------------------------------------|
| mid            | Id of the machine that the login was observed on        |
| sshd_pid_hash  | Id of the sshd privileged process associated with login |
| login_time     | Time of login                                           |
| login_username | Username used in login                                  |

#### *2.1.3. Output Data Model*

The output data generated by session tracking is represented with the following logical tables:

- login-local-descendant
- login-connection
- login-lineage

Using the data in these tables, it is possible to determine descendant processes of a given ssh login session across the environment (i.e. spanning machines). Conversely, given a process, it is possible to determine if it is a ssh login descendant and the original ssh login session for it if so.

#### *2.1.3.1. Login Local Descendant Table*

The login-local-descendant table maintains the local (i.e. on the same machine) descendant processes of each ssh login session. It has the following columns:

| Column Name   | Description                                             |
|---------------|---------------------------------------------------------|
| mid           | Id of the machine that the process was observed on      |
| sshd_pid_hash | Id of the sshd privileged process associated with login |
| pid_hash      | Id of the process                                       |

#### *2.1.3.2 Login Connections Table*

The login-connection table maintains the connections associated with ssh logins. It has the following columns:

| Column Name    | Description                                                       |
|----------------|-------------------------------------------------------------------|
| mid            | Id of the machine that the process was observed on                |
| sshd_pid_hash  | Id of the sshd privileged process associated with login           |
| login_time     | Time of login                                                     |
| login_username | The username used in login                                        |
| src_ip_addr    | Source IP address (connection was initiated from this IP address) |
| src_port       | Source port                                                       |
| dst_ip_addr    | Destination IP address (connection was made                       |

|          |                     |
|----------|---------------------|
|          | to this IP address) |
| dst_port | Destination port    |

#### 2.1.3.3 Login Lineage Table

The login-lineage table maintains the lineage of ssh login sessions. It has the following columns:

| Column Name          | Description                                                    |
|----------------------|----------------------------------------------------------------|
| mid                  | Id of the machine that ssh login was observed on               |
| sshd_pid_hash        | Id of the sshd privileged process associated with login        |
| parent_mid           | Id of the machine that the parent ssh login was observed on    |
| parent_sshd_pid_hash | Id of the sshd privileged process associated with parent login |
| origin_mid           | Id of the machine that the origin ssh login was observed on    |
| origin_sshd_pid_hash | Id of the sshd privileged process associated with origin login |

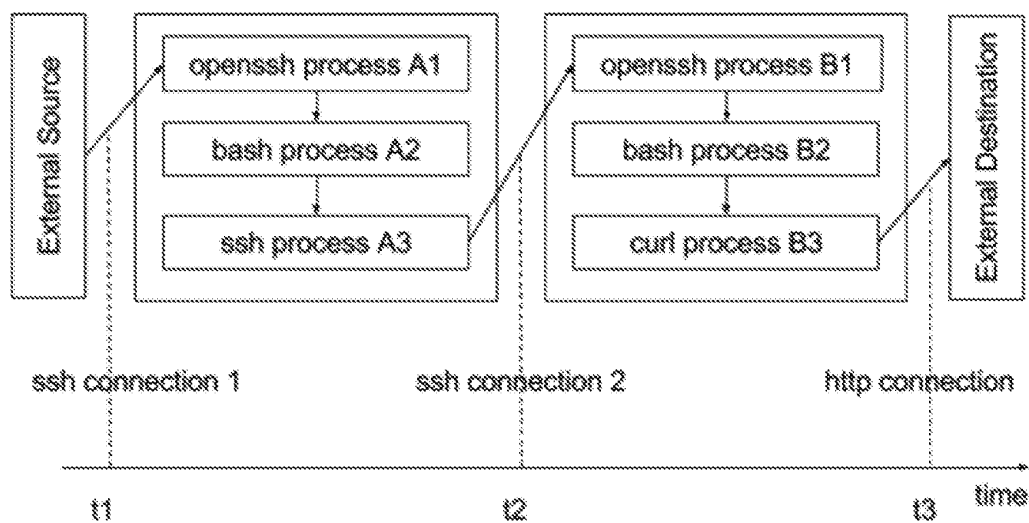
The parent\_mid and parent\_sshd\_pid\_hash columns can be null if there is no parent ssh login. In that case, (mid, sshdpid\_hash) tuple will be the same as (origin\_mid, origin\_sshd\_ppid\_hash) tuple.

### 2.3. Data Processing Flow

In various embodiments of extended user session tracking, new input data is periodically processed in a batch fashion to generate new output data. In general, batch processing allows for efficient analysis of large volumes data. However, the design can be adapted to process input data on a record-by-record fashion while maintaining the same logical data processing flow.

#### 2.3.1. Sample Scenario

The following sample scenario is used in explaining the data processing flow:



1. At time t1, there is a first ssh connection to machine A from external source IP address 1.1.1.10 and source port 10000 to destination IP address 2.2.2.20 and destination port 22 (ssh port). The source IP address is considered external because it is not associated part of the environment that is being monitored.
2. A first ssh login session LS1 is created on machine A for user X. The privileged openssh process for this login is A1.
3. Under the login session LS1, the user creates a bash shell process with pid\_hash A2.
4. At time t2, inside the bash shell process A2, the user runs ssh program under a new process A3 to login to machine B with a different username Y.
5. There is a second ssh connection to machine B from source IP address 2.2.2.20 and source port 10001 to destination IP address 2.2.2.21 and destination port 22.
6. A second ssh login session LS2 is created on machine B for user Y. The privileged openssh process for this login is B1.
7. Under the login session LS2, the user creates a bash shell process with pid\_hash B2.
8. At time t3, inside the bash shell process B2, the user runs curl command under a new process B3 to download a file from an external destination.
9. There is an https connection from source IP address 2.2.2.21 and source port 10002 to external destination IP address 3.3.3.30 port 443.

The objective is find out the original user who initiated the connection to external destination, which in this example is username X on machine A and the extended user session can be said to start with ssh login session LS1.

Based on local descendant tracking, we can make the following determinations on machine A and B already:

- A3 is a descendant of A1 and thus associated with LS1
- The connection to the external domain from machine B is initiated by B3

- B3 is a descendant of B1 and is thus associated with LS2
- Connection to the external domain is thus associated with LS2

We further establish an association between A3 and LS2 based on the fact that the LS2 was created based on a ssh connection initiated from A3. Then we can say that LS2 is a child of LS1.

If we wanted to determine the user responsible for making a connection to the external destination (e.g. if it were a known bad destination), we would first trace the process that made the connection, i.e. A3, to LS2. Then we would trace LS2 to LS1, i.e. LS1 is the origin login session for LS2. Thus the user for this connection is the user for LS1, i.e. X. In the figure above, one can visualize the tracing by following the links (in the reverse direction of arrows) from external source to A1.

In the sample scenario, we assume that both ssh connections occur in the same analysis period. However, the approach is designed such that it will work for connections and processes that are created in different time periods.

### 2.3.2. Processing Steps

The following processing steps are performed periodically. The results of a given step may be stored temporarily for use in a subsequent step.

#### *Step 1. Identify new ssh connection records*

New ssh connections started during the current time period are identified by querying the connections table. The query uses filters on the start\_time and dst\_port columns. The values of the range filter on the start\_time column will be based on the current time period. The dst\_port column will be checked against the ssh listening port(s).

By default, the ssh listening port number is 22. However, as this could vary across environments, the port(s) that openssh servers are listening to in the environment can be determined by data collection agents dynamically and used as the filter value for the dst port.

In the sample scenario, the query result will generate the following records:

| mid | start_time | pid_hash | src_ip_addr | src_port | dst_ip_addr | dst_port | prot | dir      |
|-----|------------|----------|-------------|----------|-------------|----------|------|----------|
| A   | t1         | A1       | 1.1.1.10    | 10000    | 2.2.2.20    | 22       | TCP  | Incoming |
| A   | t2         | A3       | 2.2.2.20    | 10001    | 2.2.2.21    | 22       | TCP  | Outgoing |
| B   | t2         | B1       | 2.2.2.20    | 10001    | 2.2.2.21    | 22       | TCP  | Incoming |

Note that for the connection between machine A and B, the two machines are likely to report start\_time values that are not exactly the same but close enough to be considered matching. In the above table, they are shown to be the same for simplicity.

*Step 2. Match ssh connection records*

The purpose of this step is to match the ssh connection records reported from source and destination sides for the same connection. The ssh connection records returned from the query in step 1 are matched based on the following criteria:

- The five tuples (src\_ip, dst\_ip, ip\_prot, src\_port, dst\_port) of the connection records must match
- The delta between the start time of the connections must be within a limit that would account for the worst case clock difference expected between two machines in the environment and typical connection setup latency.
- If there are multiple matches possible, then the match with the smallest time delta is chosen

Note that record from machine A for the incoming connection from the external source cannot be matched with another record as there is an agent only on the destination side for this connection.

In the sample scenario, the following matched connection records will be generated:

| src_mid | src_pid_hash | dst_mid | dst_pid_hash | dst_start_time | src_ip_addr | src_port | dst_ip_addr | dst_port |
|---------|--------------|---------|--------------|----------------|-------------|----------|-------------|----------|
| null    | null         | A       | A1           | t1             | 1.1.1.10    | 10000    | 2.2.2.20    | 22       |
| A       | A3           | B       | B1           | t2             | 2.2.2.20    | 10001    | 2.2.2.21    | 22       |

Note that the dst\_pid\_hash column values in these records will be that of the sshd privileged process associated with ssh logins.

*Step 3. Identify new login records*

New logins during the current time period are identified by querying the logins table. The query will use range filter on the login\_time column with values based on the current time period.

In the sample scenario, the query result will generate the following records:

| mid | login_time | sshd_pid_hash |
|-----|------------|---------------|
| A   | t1         | A1            |
| B   | t2         | B1            |

*Step 4. Identify new login-connection records*

In this step, matched ssh connection records created in step 2 and new login records created in step 3 are joined to create new records that will eventually be stored in the login-connection table.

The join condition is that dst\_mid of the matched connection record is equal to the mid of the login record and the dst\_pid\_hash of the matched connection record is equal to the sshd\_pid\_hash of the login record.

In the sample scenario, this step will generate the following records:

| mid | sshd_pid_hash | login_time | login_username | src_ip_addr | src_port | dst_ip_addr | dst_port |
|-----|---------------|------------|----------------|-------------|----------|-------------|----------|
| A   | A1            | t1         | X              | 1.1.1.10    | 10000    | 2.2.2.20    | 22       |
| B   | B1            | t2         | Y              | 2.2.2.20    | 10001    | 2.2.2.21    | 22       |

*Step 5. Identify login-local-descendant records in the lookback time period*

It is possible that a process that is created in a previous time period makes a ssh connection in the current analysis batch period. Even though we are not showing it in the sample scenario, consider the case where bash process A2 does not create ssh process A3 right away such that the ssh connection A3 later makes to machine B is processed in a subsequent time period than the one where A2 was processed. While processing this subsequent time period in which we see processes A3 and B1, we would want to know about A2 so that we could say B1 is associated to A3 (via ssh connection) which is associated to A2 (via process parentage) which in turn would allow us to say parent of the second ssh login is the first ssh login.

The time period for which we look back could be limited to reduce the amount of historical data we need to consider. However, this is not a requirement.

The login local descendants in the lookback time period are identified by querying the login-local-descendant table. The query uses a range filter on the login\_time column where the range is from start\_time\_of\_current\_period - lookback\_time to start\_time\_of\_current\_period.

As we are considering a single time period in the sample scenario, this step does not generate any records.

*Step 6. Identify new processes*

New processes that are started in the current time period are identified by querying the processes table. The query uses a range filter on the start\_time column with values based on the current time period.

In the sample scenario, the query result will generate the following records:

| mid | start_time | pid_hash | exe_path       | parent_pid_hash |
|-----|------------|----------|----------------|-----------------|
| A   | t1         | A1       | /usr/sbin/sshd | A0              |
| A   | t1         | A2       | /bin/bash      | A1              |
| A   | t2         | A3       | /usr/bin/ssh   | A2              |
| B   | t2         | B1       | /usr/sbin/sshd | B0              |
| B   | t2         | B2       | /bin/bash      | B1              |
| B   | t3         | B3       | /usr/bin/curl  | B2              |

*Step 7. Identify new login-local-descendant records*

In this step, we want to determine if any of the new processes in the current time period are descendants of a ssh login process and if so create records that will be at the end be stored in the login-local-descendant records for them.

In order to do so, we need to recursively follow the parent-child relationships between the processes. Either a top down or bottom up approach could be used.

In a top down approach, the ssh local descendants in the lookback period identified in step 5 combined with new ssh login processes in the current period identified in step 4 would be considered as possible ancestors for the new processes in the current period identified in step 6.

Conceptually, recursion can be considered include multiple sub-steps where new processes that are identified to be ssh local descendants in the current sub-step will be considered as ancestors for the next step. In the sample scenario, the following descendancy relationships will be established in two sub-steps:

Sub-step 1

Process A2 is a local descendant of LS1 (i.e. mid = A, sshd\_pid\_hash = A1) because it is a child of process A1 which is the login process for LS1

Process B2 is a local descendant of LS2 (i.e. mid = B, sshd\_pid\_hash = B1) because it is a child of process B1 which is the login process for LS2

Sub-step 2

Process A3 is a local descendant of LS1 because it is a child of process A2 which is associated to LS1 in sub-step 1

Process B3 is a local descendant of LS2 because it is a child of process B1 which is associated to LS2 in sub-step 1

Actual implementation for this step can use a datastore that supports recursive query capabilities or queries can be constructed to process multiple conceptual sub-steps at once.

The following records will be generated in this step:

| mid | sshd_pid_hash | pid_hash |
|-----|---------------|----------|
| A   | A1            | A1       |
| A   | A1            | A2       |
| A   | A1            | A3       |

|   |    |    |
|---|----|----|
| B | B1 | B1 |
| B | B1 | B2 |
| B | B1 | B3 |

Note that the ssh privileged processes associated with the logins are also included as they are part of the login session.

*Step 8. Identify new login-lineage records*

In this step we want to determine lineage of new ssh logins created in the current time period by associating their ssh connections to source processes that may be descendants of other ssh logins (which may have been created in the current period or previous time periods).

In order to do so, we would first try to join the new ssh login connections in the current period (identified in step 4) with the combination of the login local descendants in the lookback period (identified in step 5) and the login local descendants in the current time period (identified in step 6). This will create an adjacency relationships between child and parent logins.

In the sample scenario, we would associate the second ssh login connection to process A3 and create the following adjacency relationship between the two login sessions:

| parent_mid | parent_sshd_pid_hash | child_mid | origin_sshd_pid_hash |
|------------|----------------------|-----------|----------------------|
| A          | A1                   | B         | B1                   |

Next we use the adjacency relationships to find the original login sessions. While not shown in the sample scenario, there could be multiple ssh logins in a chain in the current time period and we would need to use a recursive approach as in step 7.

At the end of this step, the following login lineage records will be generated:

| mid | sshd_pid_hash | parent_mid | parent_sshd_pid_hash | origin_mid | origin_sshd_pid_hash |
|-----|---------------|------------|----------------------|------------|----------------------|
| A   | A1            | null       | null                 | A          | A1                   |
| B   | B1            | A          | A1                   | A          | A1                   |

*Step 9. Generate output data*

In this step, we insert the new login-connection, login-local-descendant and login-lineage records generated in steps 4, 7 and 8 to their respective output tables. This needs to be performed in a transaction manner.

# Smart Forensics

## Activity Graph

Data in platform 102 is internally organized as an activity graph.

1. Nodes of the graph are called 'Entities'
2. Activities generated by 'Entities' are modeled as edges between nodes (possibly different entity type).

I.e. every edge is an activity between 2 'Entities'. Hence the name Activity graph. Query service provides either relational views or graph views on top of this data.

## Directed edges

Edges are modeled as directed graph, i.e. they have a specific source and a destination.

## Entity Type

Entities are used as nodes in graphs. Entity type can be 1 of the following

1. Machine (agent installed)
2. Process
3. Binary
4. UID
5. IP
6. DNS
7. ListenPort
8. PType

## Common fields for all entities

1. First observed Timestamp  
Timestamp when first use of the entity was detected. First observed time does not depend on current time range
2. ClassificationType  
Machine learning based classification of the entity into clusters of similar entities. (PType, MType etc)
3. Last Observed Time  
Last observed time is does not depend on current time range.

## Activities

Examples of defined activities

1. 'Network Connected to'  
Process, IP, ListenPorts can 'Connect to' each other.  
Direction (Initiator -> server)
2. 'DNS query'

Processes, machines can do 'DNS' queries for various DNS names. A process 'P' doing a DNS query for 'www.abc.com' will be an edge from 'P' -> 'www.abc.com'

Direction (Initiator -> Query.Question)

3. 'Login'

Users can 'Login' to machines

Direction (User -> Machine)

4. 'Launched'

A 'parent' process can launch a new process.

Direction (Parent -> Child)

5. 'FKey'

FKey designates a relation between 2 entities.

Direction (Entity A has FKey to Entity B)

### Filters

Filters can be used to filter data in various ways. Filters limit the data returned by query service and do not change the data set.

Most of the data that the user is interested in is filtered on 'Activity Graph'. I.e. if an entity did not do any activity within the specified time period, it is filtered out of results. E.g. a request to show machines will be interpreted as 'Show distinct machines that were active during the specified time period'.

### FKey

FKey defines a foreign key kind of relationship between entities. If an FKey is defined, the object inherits all the 'Immutable' and 'mutable' attrs of the foreign relation. E.g.

1. Process has a FKEY called UID, MID.

This means a Process inherits all the attributes of the 2 other objects (User == UID), and (Machine == MID).

2. Machine does not have a FKEY called UID.

FKey value can be NULL for some objects. In that case Foreign relation properties is defaulted to "", or 0 etc. As an example IP can be a Machine (with an agent on), or an internal machine, vs DNS query vs no-host-name available. IP View can inherit Machine properties, if MID value is available otherwise not.

### Display value vs key

For key columns, values displayed to user may not be the same as unique internal keys and both might be different than column name. E.g. for 'Parent process'

1. Name of column is "Parent"
2. Column type is "PID\_HASH"

3. Column value is 'value for PPID\_HASH'
4. Display value might be 'value for PPID'
5. Key to be used for further QS queries is 'MID, value of PPID\_HASH'

#### **Extended vs Immutable vs Mutable attrs**

Every view has 3 sets of attributes 'Extended', 'Immutable', or 'Mutable'.

1. 'Immutable' are inherited via FKey relationship. 'Immutable' attrs do not mutate for an object over time. Thus there is 1 value over any time range.
2. 'Mutable' attrs change over time, so can have more than 1 value over a time range. Query service in general will return 1 value for them in a time range (usually latest value) along with a 'count' of actual distinct values available. UI can request a Time series if needed to visualize all the possible values over time. Mutable attrs are inherited through 'FKey' relationship.
3. 'Extended' attrs are mutable attrs that are not inherited by FKey relationships. I.e. Extended can only be used on the view that defines them.

#### **QS views**

##### *Example Input variables*

1. StartTime
2. EndTime
3. Columns projected
  - a. Columns projected can be 'immutable' or 'mutable' properties of the relations for which FKey is defined.
4. Offset
5. Limit
6. Order by
  - a. Field (or fields) to use to order
7. Aggregator function
  - a. None  
A Row data set is returned, and supports pagination (via offset, limit clauses)
  - b. Count  
A simple count of rows in the data set
  - c. Groupby  
Groups the data based on 'group by' columns. Returns 'counts' or a function on projected columns (Min, max, avg, sum, count distinct)
  - d. Distinct-Count  
Does a distinct followed by count of the dataset.
8. Filter sets

Multiple filters can be specified for different levels of sub-queries.

#### *Return values*

For 'Mutable' fields, where field has more than 1 value, a count will be returned indicating more than 1 value is available.

#### *Activity graph views*

It is possible in graph views, that dataset is too big to be returned as a Graph. In these cases, only edges can be requested and used as a table view, or only nodes can be requested and used as list of distinct entities involved in the graph.

In the first phase we are implementing table views only and will implement graph views in next phase.

#### *Dossiers*

Platform 102 supports multiple kinds of dossiers and different queries that can be issued by the dossiers are supported.

- a. Investigation Dossiers  
Investigations can be started on either 1 instance of an entity, or a set of entities. Set of entities can be either specified as an in-list, or a filter on the entity view.
- b. Comparison  
Comparison dossier requires 2 entities for comparison.
- c. Reports  
Reports can be used to put evidence from different dossiers together in 1 place.
- d. Overview  
Overview summarizes the dossiers in an investigation.

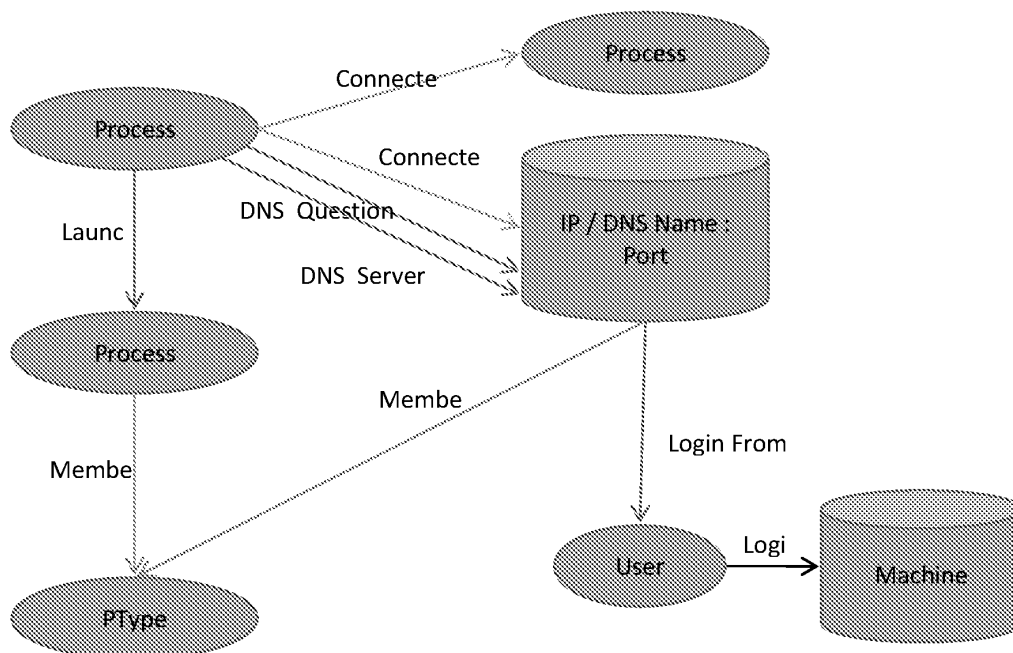
#### *Investigations*

Following is the list of example investigation questions that can be asked from platform 102.

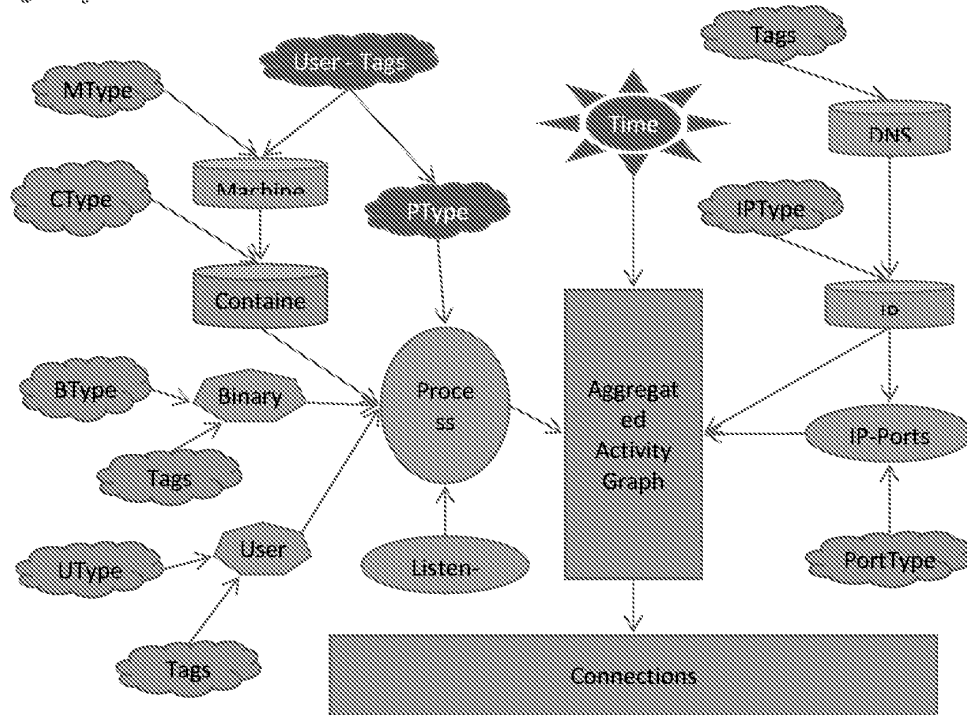
|                                                                             |                                                                                                                      |
|-----------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|
| <b>Show entities of type X, filtered on X.F</b>                             | ( Includes new entities X.FirstObserved>T)<br>Show machines in prod (Tag='prod')<br>Show processes running 'abc.exe' |
| <b>Show all distinct values of a field X.F</b>                              | Show users of a machine<br>Show machines running 'abc.exe'                                                           |
| <b>Show X.F, counts group by X.F</b>                                        | Show top commands in a machine                                                                                       |
| <b>Show TopN entities based on field values X.F (X.Memory)</b>              | Show top machines based on memory installed                                                                          |
| <b>Show TopN entities based on activity 'A' counts (X.BytesTransferred)</b> | Show top processes based on network bytes                                                                            |
| <b>Show attrs of Entity X.Instance</b>                                      | Show machine in detail                                                                                               |

|                                                                                                      |                                                                |
|------------------------------------------------------------------------------------------------------|----------------------------------------------------------------|
| Show Timeseries of (some) fields of an Entity X.Instance                                             | Show time series of ptype                                      |
| Show all 'Relation R' of X.Inst filtered by R.F                                                      | Show all processes 'launched' by process X on 'Linux' machines |
| Show all 'Relation R' of X.F=... filtered by R.F                                                     | Show all processes 'launched' by user on 'Linux' machines      |
| Show all distinct Entities 'Y' of 'Relation R' to Entity 'X.Inst'                                    | Machines, 'Connected to' IP=1.1.1.1                            |
| Show all distinct Entities 'Y' related by 'Relation R' to Entities with filter 'X.F'                 | Machines, running Process.ExePath='abc'                        |
| Show all distinct Entities 'Y' related by 'Relation R' to Entities with filter 'X.F' filtered by Y.F | Linux Machines, running Process.ExePath='abc'                  |
| Show Y.F, count of Entities 'Y' related by 'Relation R' to Entities with filter 'X.F' group by Y.F   | Group by Machine.KernelVersion, running Process.ExePath='abc'  |
| Show all distinct Entities similar to Entity 'X.Inst'                                                | Show all processes similar to process X                        |
| Show all distinct Entities similar to Entity 'X.F=...'                                               | Show all processes similar to ones running 'abc.exe'           |

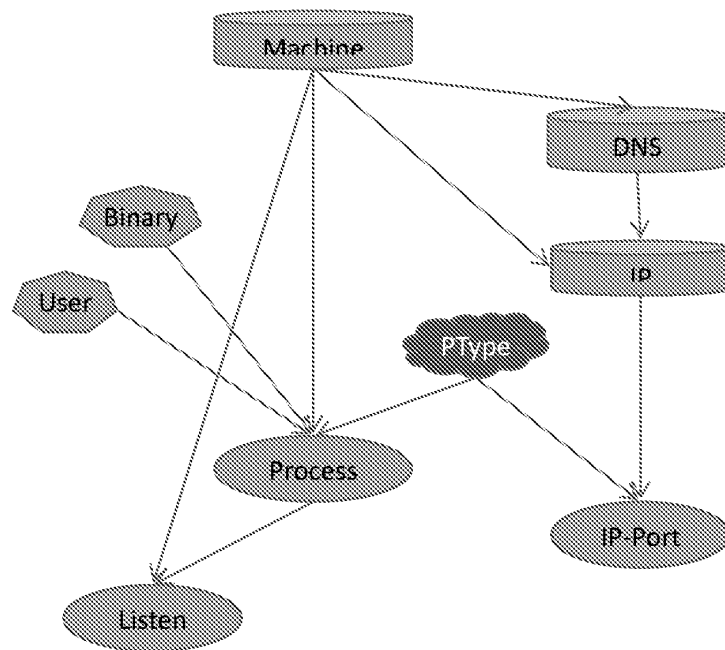
### Activity Graph

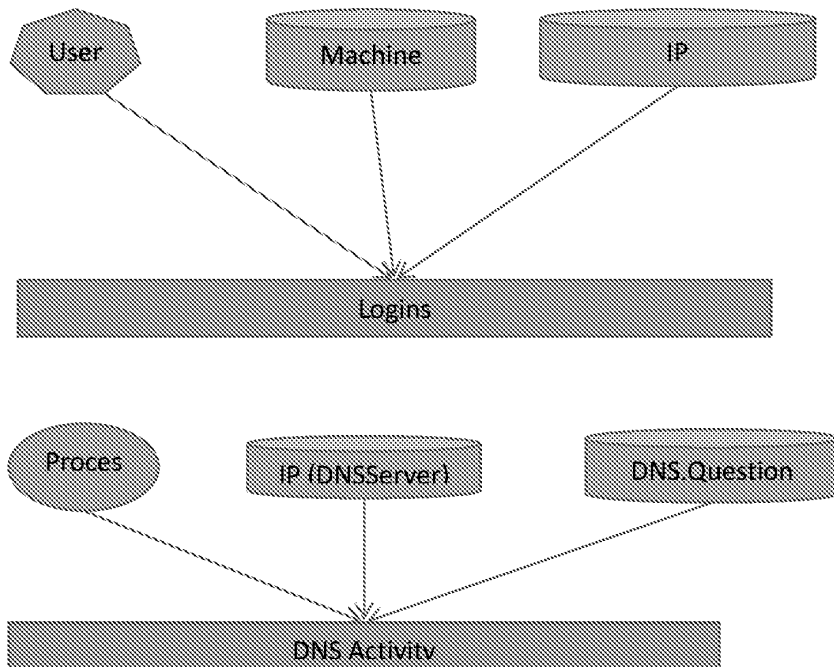


## Query structure



## Entity FKEY relations





### Alerts

Various events are generated as data comes in. Following list is supported in MVP

- A User Login
- Process start/stop
- Machine Stops/Start/Reboot
- Interface changes (Promiscus mode, Ip address, interface, up down, MAC)
- Network setting change (New Default Router, Hostname change)
- New Machine
- New Listenport
- Process Priv escalation
- A Bad login attempt
- A new DNS query from a MID
- A new CMD used on a MID
- A new IP used on a MID

## Query Service

### REST API

Dropwizard framework  
Filter based query interface  
Conjunct between different filter names  
Disjunct for the same filter names  
Dynamic join generation on server side  
Table Function and Dynamic SQL

### REST API

/describe/<namespace>/<entity>  
/query/<namespace>/<entity>?F.1&F...  
/search/<namespace>/<entity>  
/manage/<namespace>/<entity>  
GET -> SELECT  
POST -> INSERT INTO  
PUT -> UPDATE  
DELETE -> DELETE FROM

### Filter to SQL predicate

|                         |                          |
|-------------------------|--------------------------|
| /query/PROCESS_DETAILS_ | SQL                      |
| T?                      | SELECT UID               |
| F.1.N=PPID&             | FROM                     |
| F.1.V=gid&              | PROCESS_DETAILS_T        |
| F.2.N=PPID&             | WHERE                    |
| F.2.V=RID&              | (UID IN (3, 4, 5))       |
| F.3.N=UID&              | AND                      |
| F.3.V=3&                | (PPID > 1 AND PPID < 10) |
| F.3.V=4&                |                          |
| F.3.V=5&                |                          |
| ReturnAttr.1=UID        |                          |

```

/query/PROCESS_DETAILS_T? SQL
TAILS_T? SELECT UID
F.1.N=PPID& FROM
F.1.V=gt.1& PROCESS_DETAILS_T
F.1.V=lt.10& WHERE (UID IN (3, 4, 5))
F.2.N=UID& AND (PPID > 1 OR PPID < 10)
F.2.V=3&
F.2.V=4&
F.2.V=5&
ReturnAttr.1=UID

```

## Join

```

//Join(MACHINE_T,PROCESS_DETAILS_T)
Action=Describe&Version=123
IF 1.Type=SQL
IF 1.Key=SQL
E1.ReturnAttr.1=MID
E1.ReturnAttr.2=UTC_US
E1.Limit=10
E2.F.1.N=PPID
E2.F.1.V=gt.1
E2.F.1.V=lt.10
E2.F.2.N=UID
E2.F.2.V=3
E2.F.2.V=4
E2.F.2.V=5
E2.F.2.V=gt.10

SELECT MID, UTC_US
FROM lacework.MACHINE_T
WHERE MID IN (
SELECT MID
FROM PROCESS_DETAILS_T
WHERE
(UID IN (3, 4, 5) OR UID > 10)
AND
(PPID > 1 OR PPID < 10)
)
LIMIT 10

```

## Dynamic Join Generation

Dynamic join to shield UI/UX from SQL

UI/UX only sends filters

Query service generate join in SQL

Join graph created at start up time from all tables/views

Similar to FK/PK join graph by assuming common key column names (such as PID, MID, UID, etc)

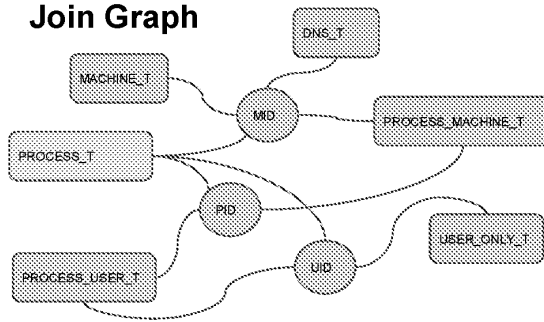
Join graph is reused for all search requests

Each request specifies target entity (fact) and its search context (dimensions)

## Searching

1. The goal is to find Join Filter(s) to reduce cardinality of the target Fact table/view.
2. Try direct link to Dim with filter  
MACHINE.MID = PROCESS.MID (PID=10)  
No need for JOIN if (MID = 10) presents
3. If nothing found at 2, try intermediate DimX, where  
Fact is connected to DimX And  
DimX is connected DimY with filter y=5
4. Finally, try 3rd degree separation from a filter Z=10  
through Fact-DimX-DimY-DimZ(z=10)

## Join Graph

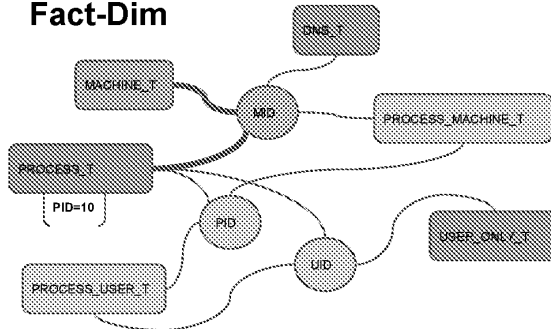


## Search (Fact-Dim)

/search/MACHINE\_T/PROCESS\_T/USER\_ONLY\_T/DNS\_T?F.1.N=PID&F.1.V=10

```
SELECT LSN, MID, MACHINE_GUID, UTC_US
FROM MACHINE_T
WHERE MID IN (
    SELECT MID FROM
    PROCESS_T
    WHERE PID = 10)
```

## Fact-Dim

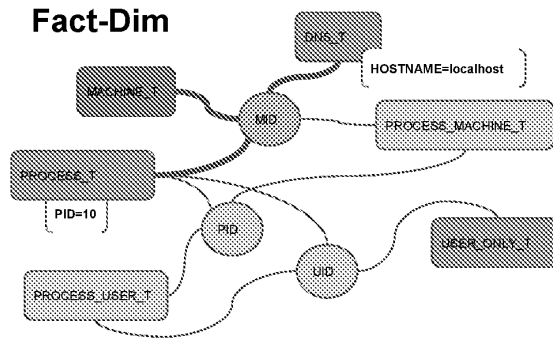


## Fact-Dim (2)

/search/MACHINE\_T/PROCESS\_T/USER\_ONLY\_T/DNS\_T?F.1.N=PID&F.1.V=10&F.2.N=HOSTNAME&F.2.V=localhost

```
SELECT LSN, MID, MACHINE_GUID, UTC_US FROM MACHINE_T
WHERE
    MID IN (SELECT MID FROM PROCESS_T WHERE PID = 10)
AND
    MID IN (SELECT MID FROM DNS_T WHERE HOSTNAME = 'localhost')
```

## Fact-Dim



## Search (Fact-Dim-Dim)

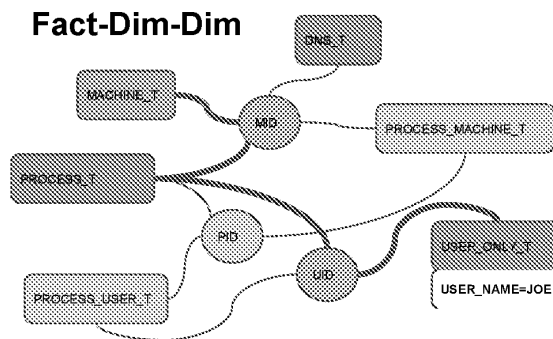
/search/MACHINE\_T/PROCESS\_T/USER\_ONLY\_T/DNS\_T?F.1.N=USER\_NAME&F.1.V=JOE

```

SELECT LSN, MID, MACHINE_GUID, UTC_US
FROM MACHINE_T
WHERE MID IN (
    SELECT MID FROM PROCESS_T
    WHERE UID IN (
        SELECT UID
        FROM USER_ONLY_T
        WHERE USER_NAME =
        'JOE')));

```

## Fact-Dim-Dim



## Search (Fact-Dim-Dim-Dim)

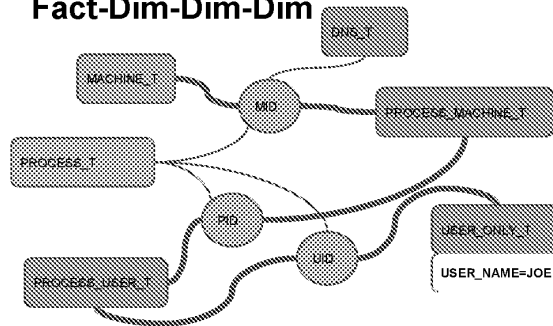
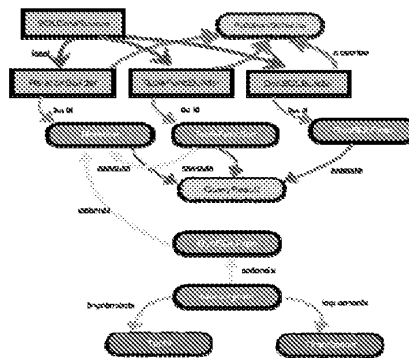
/search/MACHINE\_T/PROCESS\_USER\_T/PROCESS\_MACHINE\_T/USER\_ONLY\_T?F.1.N=USER\_NAME&F.1.V=JOE

```

SELECT LSN, MID, MACHINE_GUID, UTC_US
FROM MACHINE_T
WHERE MID IN (
    SELECT MID FROM
    PROCESS_MACHINE_T
    WHERE PID IN (
        SELECT PID FROM
        PROCESS_USER_T
        WHERE UID IN (
            SELECT UID FROM
            USER_ONLY_T
            WHERE USER_NAME
            = 'JOE')));

```

## Fact-Dim-Dim-Dim



## Dynamic SQL

Static View/Table Function is not enough

Only parameterized predicate

Can't change query structure

Templatize SQL Generation

One template for different relations based on input parameters(JSON)

Dynamic predicate push down

```

public class GenericTopNByCount extends DynamicSQL {
    public void TopN(ParamInfo params) {
        String relname = builder.getProcInfo("relation");
        String colname = builder.getProcInfo("aggrColumn");
        String alias = builder.getProcInfo("alias");
        Relation relation = builder.getRelation(params, relname);
        parents.parseFilter(relation); // predicate pushdown
        this.from(relation);
        Column column = relation.getColumn(colname);
        this.add(proj(column.getName(), type(column.getType()).as(alias),
        proj("COUNT(*)", typeNumber()).as("COUNT"));
        this.addGroupBy("1");
        this.addOrderBy("1", false);
        this.addOrderBy("2", true);
        this.setLimit(params.getLimit());
    }
}

```

## Extensible Query Interface for Dynamic Data Compositions and Filter Applications

Described herein is dynamic query generation by automatic discovery of join relations via static or dynamic filtering key specifications among composable data sets.

The extensibility of data model and filter applications is useful for data center security analytics, particularly to capture ever-changing anomalous behaviors over time. Complexity and cost of maintaining predefined schema for mixed relational and hierarchical data sets grows exponentially with increasing data volume and variety of new filter applications.

One aspect is to address extensibility of common query interface for various data models including relational, nested data structures, and graph datasets. Particularly, the user of the query interface can be agnostic to changes of existing data sets or newly created data sets. The extensible query interface also provides declarative and configurable specification for optimizing internal data generation and derivations.

Dynamic composition of filter data sets is implemented via Filter Key and Filter Key Type. A Filter Key can be defined as a list of columns and/or fields in nested structure (such as JSON). Instances of the same Filter Key Type can be formed as an Implicit Join Group. Same instance of Filter Key can participate in different Implicit Join Groups. List of relationships among all possible Implicit Join Groups is represented as Join Graph for entire search space to create the final filter data set by traversing edges and producing Join Path(s).

### Extensibility Example

There are two aspects of extensibility in terms of dynamic filter applications - Filter Key in Data Set and Filter Data Set itself. An example implementation of a new data set can be defined as follows:

Step 1 - Product manager collects customer feedback and defines new requirements for:

- New data transformations or aggregation from existing data sets.
- Data visualization for the newly defined data set

Step 2 - New Data Set (Card) defined in Query Service Catalog

- For reusability and maintainability, one external data card might be decomposed to multiple (nested) internal cards.
- Define filter key(s) for each newly added cards (including internal ones)

Step 3 - UI Developer visualize the new data set in dossier template(s)

- Same external card can be customized via parameters, ordering, or various mappings of external data fields (columns)
- One external data card can be used in multiple dossier templates
- The same external card can be referenced more than once in the same dossier template

Example extensibility using Step 2 by Card189 and User Tracking filter data set.

### Binary Information (a.k.a Card189)

This data set shows a list of binary information running by users within the period of [StartTimeRange, EndTimeRange]

#### Card189 Spec

```
"Card189" : {
  "props" : {
    "question": "Hostname, Binary, and Username",
    "scope" : "external"
  },
  "parameters": [
    {"name": "StartTimeRange", "type": "TIMESTAMP"},
    {"name": "EndTimeRange", "type": "TIMESTAMP"}
  ],
  "sources": [
    {"name": "MVIEW_INTERNAL.ENTITY_VIEW_T", "alias": "P"},
    {"name": "ProcessClusterFilters", "alias": "Act"}
  ],
  "returns" : [
    {"field": "P.EXE_PATH", "alias": "EXE_PATH"},
    {"field": "P.USERNAME", "props": {"utype": "USERNAME"}},
    {"field": "P.PROPS_PROCESS:SSH_SESSION:props:username", "alias": "ORIGIN_USER"},
    {"field": "P.HOSTNAME", "props": {"utype": "HOSTNAME"}},
    {"expr": "COUNT(DISTINCT #{P.MID}, #{P.PID_HASH})", "type": "NUMBER", "alias":
"NUM_PROCESSES"}
  ],
  "fkeys" : [
    {"type": "PID_KEY", "fieldRefs": ["P.MID", "P.PID_HASH"]}
  ],
  "joins" : [
    {"type": "implicit", "left": "PID_KEY", "right": "Act.P_PID_KEY"}
  ],
  "groups" : [
    {"field": "1"}, {"field": "2"}, {"field": "3"}, {"field": "4"}
  ],
  "filters" : [
    {"type": "ge", "field": "START_TIME", "value": "${StartTimeRange}"},
    {"type": "le", "field": "END_TIME", "value": "${EndTimeRange}"},
    {"type": "eq", "field": "ENTITY_TYPE", "value": "Process"},
    {"type": "ne", "field": "PID_HASH", "value": "0"}
  ],
  "filterMap" : {
    "P.PROPS_PROCESS:SSH_SESSION:props:username" : ["SSH_ORIGIN_USERNAME"]
  }
}
```

## Description

This card is implemented by extending existing (base) card MVIEW\_INTERNAL.ENTITY\_VIEW\_T (see the **returns** field) with explicit filters on ENTITY\_TYPE (see **filters** field). It also adds dynamic filters via **fkeys** (HOSTNAME\_KEY) and **implicit** join.

### User Tracking Filter Data Set

A definition of data sets is composable/reusable/extensible whether it is relational or graph data sets. In this case, the User Tracking PolyGraph are generated as graph data sets (nodes/edges). Like other polygraph(s), User Tracking is an external data sets and visualized as a graph by nodes and edges. It can also be used as filter data sets for other cards via the CID (Cluster ID) field. Since User Tracking is defined after Card189, it is a good example to show the extensibility of ProcessClusterFilters without changing Card233 (and all other existing cards having ProcessClusterFilters in the sources field).

### User Tracking Membership Data Sets

Following User Tracking data sets are added as new data sources for ProcessClusterFilters. Corresponding CID fields (UserTrackingIpAddress\_CID, UserTrackingUser\_CID, and UserTrackingDnsSep\_CID) are also added as external filters under ProcessClusterFilters.

```
"UserTrackingMemberOfIpAddress_MV" : {
  "type": "Function",
  "sources" : [
    {"name" : "ClusterMemberOfIpAddress_MV"}
  ],
  "parameters": [
    {"name" : "ClusterType", "value" : "UserTracking"}
  ]
},
"UserTrackingMemberOfUser_MV" : {
  "type": "Function",
  "sources" : [
    {"name" : "ClusterMemberOfUsername_MV"}
  ],
  "parameters": [
    {"name" : "ClusterType", "value" : "UserTracking"}
  ]
},
"UserTrackingMemberOfDnsSep_MV" : {
  "type": "Function",
  "sources" : [
    {"name" : "ClusterMemberOfDnsSep_MV"}
  ],
  "parameters": [
    {"name" : "ClusterType", "value" : "UserTracking"}
  ]
}
```

```
}
```

### REST API Examples

Two example shows extensibility of dynamic filter application.

Example 1- All existing filters applicable to new card (Card189)

Example 2 - New filter data set (UserTracking) applicable to all existing cards.

#### Example 1 - Card189 filtered by External DNS Name

```
curl -ss -H "Content-Type: application/json" -X POST -d '{"filters":[{"type":"like",  
"field":"ProcessClusterFilters.P2DNS_HOSTNAME", "value":"*google*"}], "paramInfo":  
[{"name": "StartTimeRange", "value": "6/17/2017"}, {"name": "EndTimeRange", "value":  
"6/18/2017"}]}'  
"localhost:8080/card/query/Card189?DCGUID=STAGING_CDB_STAGING_00_DEED95AE3C  
794A7FF1D0B"  
{  
  "data": [{"EXE_PATH": "/usr/lib/jvm/java-8-  
oracle/jre/bin/java", "USERNAME": "root", "ORIGIN_USER": "root", "HOSTNAME": "staging_worker  
5.lacework.internal", "NUM_PROCESSES": "1"}], "ok": true}
```

#### Query Service Log for Join Path search (Breadth First)

```
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-  
[Link<M2DNS_MID_KEY>]->M2DNS  
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-  
[Link<I_MID_KEY>]->I  
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-  
[Link<DNS_MID_KEY>]->DNS  
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-  
[Link<Mt2CtMachine_MID_KEY>]->Mt2CtMachine  
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-  
[Link<MtypeMachine_MID_KEY>]->MtypeMachine  
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-  
[Link<UserLoginMachine_MID_KEY>]->UserLoginMachine  
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-  
[Link<P2IPA_PID_KEY>]->P2IPA  
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-  
[Link<P2DNS_PID_KEY>]->P2DNS  
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-  
[Link<EVENT_PID_KEY>]->Event  
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-  
[Link<UserLoginUser_KEY>]->UserLoginUser  
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-  
[Link<M2DNS_MID_KEY>]->M2DNS - M2DNS-[Link<I_MID_KEY>]->I}  
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-  
[Link<M2DNS_MID_KEY>]->M2DNS - M2DNS-[Link<DNS_MID_KEY>]->DNS}  
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-  
[Link<M2DNS_MID_KEY>]->M2DNS - M2DNS-[Link<Mt2CtMachine_MID_KEY>]->Mt2CtMachine}  
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-  
[Link<M2DNS_MID_KEY>]->M2DNS - M2DNS-[Link<MtypeMachine_MID_KEY>]->MtypeMachine}  
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-  
[Link<M2DNS_MID_KEY>]->M2DNS - M2DNS-[Link<UserLoginMachine_MID_KEY>]->UserLoginMachine}  
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-  
[Link<M2DNS_MID_KEY>]->M2DNS - M2DNS-[Link<MtypeDnsName_KEY>]->MtypeDnsName}  
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-  
[Link<M2DNS_MID_KEY>]->M2DNS - M2DNS-[Link<CtypeDnsSep_KEY_2>]->CtypeDnsSep}  
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<I_MID_KEY>]-  
>I - I-[Link<DNS_MID_KEY>]->DNS}
```

```

DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<I_MID_KEY>]->I - I-[Link<M2CtMachine_MID_KEY>]->M2CtMachine}
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<I_MID_KEY>]->I - I-[Link<MtypeMachine_MID_KEY>]->MtypeMachine}
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<I_MID_KEY>]->I - I-[Link<UserLoginMachine_MID_KEY>]->UserLoginMachine}
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<I_MID_KEY>]->I - I-[Link<P2IPA_IPA_KEY>]->P2IPA}
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<DNS_MID_KEY>]->DNS - DNS-[Link<M2CtMachine_MID_KEY>]->M2CtMachine}
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<DNS_MID_KEY>]->DNS - DNS-[Link<MtypeMachine_MID_KEY>]->MtypeMachine}
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<DNS_MID_KEY>]->DNS - DNS-[Link<UserLoginMachine_MID_KEY>]->UserLoginMachine}
DEBUG [2017-06-19 01:06:46,251] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<DNS_MID_KEY>]->DNS - DNS-[Link<P2DNS_HOSTNAME_KEY>]->P2DNS}
DEBUG [2017-06-19 01:06:46,252] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<M2CtMachine_MID_KEY>]->M2CtMachine - M2CtMachine-[Link<MtypeMachine_MID_KEY>]->MtypeMachine}
DEBUG [2017-06-19 01:06:46,252] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<M2CtMachine_MID_KEY>]->M2CtMachine - M2CtMachine-[Link<UserLoginMachine_MID_KEY>]->UserLoginMachine}
DEBUG [2017-06-19 01:06:46,252] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<MtypeMachine_MID_KEY>]->MtypeMachine - MtypeMachine-[Link<UserLoginMachine_MID_KEY>]->UserLoginMachine}
DEBUG [2017-06-19 01:06:46,252] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA_PID_KEY>]->P2IPA - P2IPA-[Link<P2DNS_PID_KEY>]->P2DNS}
DEBUG [2017-06-19 01:06:46,252] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA_PID_KEY>]->P2IPA - P2IPA-[Link<EVENT_PID_KEY>]->Event}
DEBUG [2017-06-19 01:06:46,252] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA_PID_KEY>]->P2IPA - P2IPA-[Link<MtypeIpAddress_KEY>]->MtypeIpAddress}
DEBUG [2017-06-19 01:06:46,252] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA_PID_KEY>]->P2IPA - P2IPA-[Link<MtypeIpSep_KEY>]->MtypeIpSep}
DEBUG [2017-06-19 01:06:46,252] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA_PID_KEY>]->P2IPA - P2IPA-[Link<PtypeIpAddress_KEY>]->PtypeIpAddress}
DEBUG [2017-06-19 01:06:46,252] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA_PID_KEY>]->P2IPA - P2IPA-[Link<PtypeIpSep_KEY>]->PtypeIpSep}
DEBUG [2017-06-19 01:06:46,252] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA_PID_KEY>]->P2IPA - P2IPA-[Link<IntPConnIpAddress_KEY>]->IntPConnIpAddress}
DEBUG [2017-06-19 01:06:46,252] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA_PID_KEY>]->P2IPA - P2IPA-[Link<CtypeIpAddress_KEY>]->CtypeIpAddress}
DEBUG [2017-06-19 01:06:46,252] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA_PID_KEY>]->P2IPA - P2IPA-[Link<CtypeIpSep_KEY>]->CtypeIpSep}
DEBUG [2017-06-19 01:06:46,252] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA_PID_KEY>]->P2IPA - P2IPA-[Link<UserLoginIpAddress_KEY>]->UserLoginIpAddress}
DEBUG [2017-06-19 01:06:46,252] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add candidate path - {MVP-[Link<P2DNS_PID_KEY>]->P2DNS}
DEBUG [2017-06-19 01:06:46,252] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<EVENT_PID_KEY>]->Event - Event-[Link<P2DNS_PID_KEY>]->P2DNS}
DEBUG [2017-06-19 01:06:46,253] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<M2DNS_MID_KEY>]->M2DNS - M2DNS-[Link<MtypeDnsName_KEY>]->MtypeDnsName - MtypeDnsName-[Link<CtypeDnsSep_KEY_2>]->CtypeDnsSep}
DEBUG [2017-06-19 01:06:46,253] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<M2DNS_MID_KEY>]->M2DNS - M2DNS-[Link<CtypeDnsSep_KEY_2>]->CtypeDnsSep - CtypeDnsSep-[Link<P2DNS_HOSTNAME_KEY_C>]->P2DNS}

```

#### Dynamic SQL Generation

```

WITH GRAPH_EDGE_VIEW AS (
  WITH GRAPH_EDGE_VIEW_8 AS (
    SELECT ID, SRC_TYPE, SRC_KEY, DST_TYPE, DST_KEY, TYPE, PROPS, START_TIME, END_TIME
    FROM GRAPH_INTERNAL.EDGE_T
    WHERE START_TIME >= TO_TIMESTAMP('6/17/2017')
    AND END_TIME <= TO_TIMESTAMP('6/18/2017')

```

```

)
SELECT ID, SRC_TYPE, SRC_KEY, DST_TYPE, DST_KEY, TYPE, PROPS, START_TIME, END_TIME
FROM GRAPH_EDGE_VIEW_8
),
Card189 AS (
WITH P AS (
SELECT START_TIME, END_TIME, ENTITY_TYPE, MID, PID_HASH, USERNAME, EXE_PATH, HOSTNAME,
PROPS_PROCESS
FROM MVIEW_INTERNAL.ENTITY_VIEW_T
),
Act_MVP AS (
WITH ENTITY_VIEW_T_42 AS (
SELECT START_TIME, END_TIME, ENTITY_TYPE, MID, PID_HASH, USERNAME, EXE_PATH, HOSTNAME, CONTAINER,
NET_STATS,
CLUSTER_MAP, TAGS, EVENT_MAP, PROPS_MACHINE, PROPS_PROCESS, PROPS_USER, PROPS_EXE_PATH,
PROPS_CONTAINER
FROM MVIEW_INTERNAL.ENTITY_VIEW_T
)
SELECT ENTITY_VIEW_T_42.MID MID, ENTITY_VIEW_T_42.PID_HASH PID_HASH, ENTITY_VIEW_T_42.HOSTNAME
HOSTNAME,
ENTITY_VIEW_T_42.USERNAME USERNAME, ENTITY_VIEW_T_42.EXE_PATH EXE_PATH,
ENTITY_VIEW_T_42.CONTAINER CONTAINER,
ENTITY_VIEW_T_42.CLUSTER_MAP "CLUSTER_MAP", ENTITY_VIEW_T_42.TAGS "TAGS",
ENTITY_VIEW_T_42.PROPS_PROCESS "PROPS_PROCESS"
FROM ENTITY_VIEW_T_42
WHERE ENTITY_VIEW_T_42.ENTITY_TYPE = 'Process'
AND ENTITY_VIEW_T_42.START_TIME >= TO_TIMESTAMP('6/17/2017')
AND ENTITY_VIEW_T_42.END_TIME <= TO_TIMESTAMP('6/18/2017')
),
Act_P2DNS AS (
WITH ProcessToExternalDNSView_6 AS (
SELECT DISTINCT SRC_KEY:mid::BIGINT MID,
COALESCE(SRC_KEY:pid_hash, 0)::BIGINT PID_HASH, DST_KEY:hostname::VARCHAR HOSTNAME
FROM GRAPH_EDGE_VIEW
WHERE TYPE = 'ConnectedTo'
AND SRC_TYPE IN ('Process', 'MachineUc')
AND DST_KEY:hostname <> ''
)
SELECT MID, PID_HASH, HOSTNAME
FROM ProcessToExternalDNSView_6
)
SELECT P.EXE_PATH EXE_PATH, P.USERNAME USERNAME, P.PROPS_PROCESS:SSH_SESSION:props:username
ORIGIN_USER,
P.HOSTNAME HOSTNAME, COUNT(DISTINCT P.MID, P.PID_HASH) NUM_PROCESSES
FROM P
WHERE P.START_TIME >= TO_TIMESTAMP('6/17/2017')
AND P.END_TIME <= TO_TIMESTAMP('6/18/2017')
AND P.ENTITY_TYPE = 'Process'
AND P.PID_HASH <> 0
AND (P.MID, P.PID_HASH) IN (
SELECT Act_MVP.MID, Act_MVP.PID_HASH FROM Act_MVP
WHERE (Act_MVP.MID, Act_MVP.PID_HASH) IN (
SELECT Act_P2DNS.MID, Act_P2DNS.PID_HASH
FROM Act_P2DNS
WHERE Act_P2DNS.HOSTNAME LIKE ?{P0} ESCAPE '\')
)
)
GROUP BY 1, 2, 3, 4
)
SELECT Card189.EXE_PATH EXE_PATH, Card189.USERNAME USERNAME, Card189.ORIGIN_USER ORIGIN_USER,
Card189.HOSTNAME HOSTNAME,

```

Card189.NUM\_PROCESSES NUM\_PROCESSES FROM Card189 LIMIT 5000

;

#### Example 2 - Card189 filtered by User Tracking cluster

```
curl -ss -H "Content-Type: application/json" -X POST -d
'{"filters":[{"type":"eq","value":"51","field":"ProcessClusterFilters.UserTrackingDnsSep_CID"}],
"paramInfo": [{"name": "StartTimeRange","value": "6/17/2017"}, {"name": "EndTimeRange","value": "6/18/2017"}]}'
"localhost:8080/card/query/Card189?DCGUID=STAGING_CDB_STAGING_00_DEED95AE3C794A7FF1D0B"
{"data":[{"EXE_PATH":"/usr/bin/python2.7","USERNAME":"root","ORIGIN_USER":"root","HOSTNAME":"staging_master0.lacework.internal","NUM_PROCESSES":"5"},{"EXE_PATH":"/bin/bash","USERNAME":"vikram_prod","ORIGIN_USER":"vikram_prod","HOSTNAME":"staging_worker4.lacework.internal","NUM_PROCESSES":"1"}],"ok":true}
```

#### Query Service Log for Join Path search (Breadth First)

```
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-
[Link<M2DNS_MID_KEY>]->M2DNS
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-
[Link<I_MID_KEY>]->I
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-
[Link<DNS_MID_KEY>]->DNS
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-
[Link<Mt2CtMachine_MID_KEY>]->Mt2CtMachine
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-
[Link<MtypeMachine_MID_KEY>]->MtypeMachine
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-
[Link<UserLoginMachine_MID_KEY>]->UserLoginMachine
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-
[Link<P2IPA_PID_KEY>]->P2IPA
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-
[Link<P2DNS_PID_KEY>]->P2DNS
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-
[Link<EVENT_PID_KEY>]->Event
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-
[Link<UserLoginUser_KEY>]->UserLoginUser
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add starting edge - MVP-
[Link<UserTrackingUser_KEY>]->UserTrackingUser
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-
[Link<M2DNS_MID_KEY>]->M2DNS - M2DNS-[Link<I_MID_KEY>]->I}
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-
[Link<M2DNS_MID_KEY>]->M2DNS - M2DNS-[Link<DNS_MID_KEY>]->DNS}
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-
[Link<M2DNS_MID_KEY>]->M2DNS - M2DNS-[Link<Mt2CtMachine_MID_KEY>]->Mt2CtMachine}
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-
[Link<M2DNS_MID_KEY>]->M2DNS - M2DNS-[Link<MtypeMachine_MID_KEY>]->MtypeMachine}
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-
[Link<M2DNS_MID_KEY>]->M2DNS - M2DNS-[Link<UserLoginMachine_MID_KEY>]->UserLoginMachine}
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-
[Link<M2DNS_MID_KEY>]->M2DNS - M2DNS-[Link<MtypeDnsName_KEY>]->MtypeDnsName}
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-
[Link<M2DNS_MID_KEY>]->M2DNS - M2DNS-[Link<CtypeDnsSep_KEY_2>]->CtypeDnsSep}
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<I_MID_KEY>]-
>I - I-[Link<DNS_MID_KEY>]->DNS}
DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<I_MID_KEY>]-
>I - I-[Link<Mt2CtMachine_MID_KEY>]->Mt2CtMachine}
```

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<I\_MID\_KEY>]->I - I-[Link<MtypeMachine\_MID\_KEY>]->MtypeMachine}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<I\_MID\_KEY>]->I - I-[Link<UserLoginMachine\_MID\_KEY>]->UserLoginMachine}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<I\_MID\_KEY>]->I - I-[Link<P2IPA\_IPA\_KEY>]->P2IPA}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<DNS\_MID\_KEY>]->DNS - DNS-[Link<Mt2CtMachine\_MID\_KEY>]->Mt2CtMachine}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<DNS\_MID\_KEY>]->DNS - DNS-[Link<MtypeMachine\_MID\_KEY>]->MtypeMachine}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<DNS\_MID\_KEY>]->DNS - DNS-[Link<UserLoginMachine\_MID\_KEY>]->UserLoginMachine}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<DNS\_MID\_KEY>]->DNS - DNS-[Link<P2DNS\_HOSTNAME\_KEY>]->P2DNS}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<Mt2CtMachine\_MID\_KEY>]->Mt2CtMachine - Mt2CtMachine-[Link<MtypeMachine\_MID\_KEY>]->MtypeMachine}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<Mt2CtMachine\_MID\_KEY>]->Mt2CtMachine - Mt2CtMachine-[Link<UserLoginMachine\_MID\_KEY>]->UserLoginMachine}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<MtypeMachine\_MID\_KEY>]->MtypeMachine - MtypeMachine-[Link<UserLoginMachine\_MID\_KEY>]->UserLoginMachine}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA\_PID\_KEY>]->P2IPA - P2IPA-[Link<P2DNS\_PID\_KEY>]->P2DNS}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA\_PID\_KEY>]->P2IPA - P2IPA-[Link<EVENT\_PID\_KEY>]->Event}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA\_PID\_KEY>]->P2IPA - P2IPA-[Link<MtypeIpAddress\_KEY>]->MtypeIpAddress}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA\_PID\_KEY>]->P2IPA - P2IPA-[Link<MtypeIpSep\_KEY>]->MtypeIpSep}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA\_PID\_KEY>]->P2IPA - P2IPA-[Link<PtypeIpAddress\_KEY>]->PtypeIpAddress}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA\_PID\_KEY>]->P2IPA - P2IPA-[Link<PtypeIpSep\_KEY>]->PtypeIpSep}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA\_PID\_KEY>]->P2IPA - P2IPA-[Link<IntPConnIpAddress\_KEY>]->IntPConnIpAddress}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA\_PID\_KEY>]->P2IPA - P2IPA-[Link<CtypeIpAddress\_KEY>]->CtypeIpAddress}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA\_PID\_KEY>]->P2IPA - P2IPA-[Link<CtypeIpSep\_KEY>]->CtypeIpSep}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA\_PID\_KEY>]->P2IPA - P2IPA-[Link<UserLoginIpAddress\_KEY>]->UserLoginIpAddress}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2IPA\_PID\_KEY>]->P2IPA - P2IPA-[Link<UserTrackingIpAddress\_KEY>]->UserTrackingIpAddress}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2DNS\_PID\_KEY>]->P2DNS - P2DNS-[Link<EVENT\_PID\_KEY>]->Event}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2DNS\_PID\_KEY>]->P2DNS - P2DNS-[Link<PtypeDnsSep\_KEY>]->PtypeDnsSep}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2DNS\_PID\_KEY>]->P2DNS - P2DNS-[Link<CtypeDnsSep\_KEY>]->CtypeDnsSep}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<P2DNS\_PID\_KEY>]->P2DNS - P2DNS-[Link<UserTrackingDnsSep\_KEY>]->UserTrackingDnsSep}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<UserLoginUser\_KEY>]->UserLoginUser - UserLoginUser-[Link<UserTrackingUser\_KEY>]->UserTrackingUser}

DEBUG [2017-06-19 00:30:33,239] net.lacework.querybuilder.core.implicitfilter.EntityFilters: new path - {MVP-[Link<M2DNS\_MID\_KEY>]->M2DNS - M2DNS-[Link<MtypeDnsName\_KEY>]->MtypeDnsName - MtypeDnsName-[Link<CtypeDnsSep\_KEY\_2>]->CtypeDnsSep}

**DEBUG [2017-06-19 00:30:33,240] net.lacework.querybuilder.core.implicitfilter.EntityFilters: add candidate path - {MVP-[Link<P2DNS\_PID\_KEY>]->P2DNS - P2DNS-[Link<UserTrackingDnsSep\_KEY>]->UserTrackingDnsSep}**

### Dynamic SQL

```
WITH GRAPH_EDGE_VIEW AS (
  WITH GRAPH_EDGE_VIEW_8 AS (
    SELECT ID, SRC_TYPE, SRC_KEY, DST_TYPE, DST_KEY, TYPE, PROPS, START_TIME, END_TIME
    FROM GRAPH_INTERNAL.EDGE_T
    WHERE START_TIME >= TO_TIMESTAMP('6/17/2017')
    AND END_TIME <= TO_TIMESTAMP('6/18/2017')
  )
  SELECT ID, SRC_TYPE, SRC_KEY, DST_TYPE, DST_KEY, TYPE, PROPS, START_TIME, END_TIME
  FROM GRAPH_EDGE_VIEW_8
),
MViewKeyClusterMap AS (
  WITH MViewKeyClusterMap_72 AS (
    SELECT DISTINCT ENTITY_TYPE, MID, PID_HASH, USERNAME, EXE_PATH, IP_ADDR, INTERNAL, HOSTNAME,
    CONTAINER,
    PORT, PROTOCOL, CLUSTER_MAP
    FROM MVIEW_INTERNAL.ENTITY_VIEW_T
    WHERE START_TIME >= TO_TIMESTAMP('6/17/2017')
    AND END_TIME <= TO_TIMESTAMP('6/18/2017')
  )
  SELECT ENTITY_TYPE, MID, PID_HASH, USERNAME, EXE_PATH, IP_ADDR, INTERNAL, HOSTNAME, CONTAINER, PORT,
  PROTOCOL, CLUSTER_MAP
  FROM MViewKeyClusterMap_72
),
Card189 AS (
  WITH P AS (
    SELECT START_TIME, END_TIME, ENTITY_TYPE, MID, PID_HASH, USERNAME, EXE_PATH, HOSTNAME,
    PROPS_PROCESS
    FROM MVIEW_INTERNAL.ENTITY_VIEW_T
  ),
  Act_MVP AS (
    WITH ENTITY_VIEW_T_42 AS (
      SELECT START_TIME, END_TIME, ENTITY_TYPE, MID, PID_HASH, USERNAME, EXE_PATH, HOSTNAME, CONTAINER,
      NET_STATS,
      CLUSTER_MAP, TAGS, EVENT_MAP, PROPS_MACHINE, PROPS_PROCESS, PROPS_USER, PROPS_EXE_PATH,
      PROPS_CONTAINER
      FROM MVIEW_INTERNAL.ENTITY_VIEW_T
    )
    SELECT ENTITY_VIEW_T_42.MID MID, ENTITY_VIEW_T_42.PID_HASH PID_HASH, ENTITY_VIEW_T_42.HOSTNAME
    HOSTNAME,
    ENTITY_VIEW_T_42.USERNAME USERNAME, ENTITY_VIEW_T_42.EXE_PATH EXE_PATH,
    ENTITY_VIEW_T_42.CONTAINER CONTAINER,
    ENTITY_VIEW_T_42.CLUSTER_MAP "CLUSTER_MAP", ENTITY_VIEW_T_42.TAGS "TAGS",
    ENTITY_VIEW_T_42.PROPS_PROCESS "PROPS_PROCESS"
    FROM ENTITY_VIEW_T_42
    WHERE ENTITY_VIEW_T_42.ENTITY_TYPE = 'Process'
    AND ENTITY_VIEW_T_42.START_TIME >= TO_TIMESTAMP('6/17/2017')
    AND ENTITY_VIEW_T_42.END_TIME <= TO_TIMESTAMP('6/18/2017')
  ),
  Act_P2DNS AS (
    WITH ProcessToExternalDNSView_6 AS (
      SELECT DISTINCT SRC_KEY:mid::BIGINT MID,
      COALESCE(SRC_KEY:pid_hash, 0)::BIGINT PID_HASH, DST_KEY:hostname::VARCHAR HOSTNAME
      FROM GRAPH_EDGE_VIEW
      WHERE TYPE = 'ConnectedTo'
      AND SRC_TYPE IN ('Process', 'MachineUc') AND DST_KEY:hostname <> ''
    )
    SELECT MID, PID_HASH, HOSTNAME
    FROM ProcessToExternalDNSView_6
  ),
```

```

Act_UserTrackingDnsSep AS (
  WITH UserTrackingMemberOfDnsSep_MV_70 AS (
    SELECT DISTINCT CLUSTER_MAP:UserTracking::VARCHAR CID, HOSTNAME, PORT, PROTOCOL
    FROM MViewKeyClusterMap
    WHERE ENTITY_TYPE = 'DnsSep'
    AND CLUSTER_MAP:UserTracking IS NOT NULL
  )
  SELECT CID, HOSTNAME
  FROM UserTrackingMemberOfDnsSep_MV_70
)
SELECT
  P.EXE_PATH EXE_PATH, P.USERNAME USERNAME, P.PROPS_PROCESS:SSH_SESSION:props:username
  ORIGIN_USER,
  P.HOSTNAME HOSTNAME, COUNT(DISTINCT P.MID, P.PID_HASH) NUM_PROCESSES
FROM P
WHERE P.START_TIME >= TO_TIMESTAMP('6/17/2017')
AND P.END_TIME <= TO_TIMESTAMP('6/18/2017')
AND P.ENTITY_TYPE = 'Process'
AND P.PID_HASH <> 0
AND (P.MID, P.PID_HASH) IN (
  SELECT Act_MVP.MID, Act_MVP.PID_HASH
  FROM Act_MVP
  WHERE (Act_MVP.MID, Act_MVP.PID_HASH) IN (
    SELECT Act_P2DNS.MID, Act_P2DNS.PID_HASH
    FROM Act_P2DNS
    WHERE (Act_P2DNS.HOSTNAME) IN (
      SELECT Act_UserTrackingDnsSep.HOSTNAME
      FROM Act_UserTrackingDnsSep
      WHERE Act_UserTrackingDnsSep.CID = ?{P0}
    )
  )
)
GROUP BY 1, 2, 3, 4
)
SELECT Card189.EXE_PATH EXE_PATH, Card189.USERNAME USERNAME, Card189.ORIGIN_USER ORIGIN_USER,
Card189.HOSTNAME HOSTNAME, Card189.NUM_PROCESSES NUM_PROCESSES
FROM Card189 LIMIT 5000

```

# Insider Threat Detection

Two of the crucial roles for a security ops team are: Detecting breaches and Investigating breaches.

One way to address the two problems is to model the user behavior with respect to

- The user entrypoint - IpAddress/Country/City/Geoip a user logs in from
- Binaries/executables/process/server a user launches
- Any internal server it communicates to
- External domains or IpAddresses to which a user communicates/transfers data to (via a process)

In an event of a security breach, answering these questions becomes very important. Also, it is important to provide an end-to-path in order to investigate any breach/attack.

## Concepts

**User activity (Pre and post login):**

### **Login:**

In a typical everyday working of datacenter:

A user logs in,  
run some processes and  
logs out (or stay logged in).

For a normal day to day operations, a user will login:

from the same set of Ip Addresses or  
From the the same city or  
From a known geographical location with respect to that user(historically known).

Any deviation from norm indicates a change in login behavior. This does not necessarily mean a breach.

### **Inside the network (post login):**

Now, Once logged in, a user may do one of the following:

1. Launch a binary/script,  
which can talk to other servers inside the datacenter.  
E.g.: running "curl" to get data from some service running inside datacenter.
2. POST/Transfer data to some external server.  
E.g.: send data to some amazon box outside the datacenter
3. Change privilege.  
E.g.: become 'root' or some other user.

After this the user can continue doing any of the above any number of times

4. Login to some other machine inside the Datacenter.

After this the user can continue doing any of the above any number of times

*One such scenario can be:*

1. Say, from IP\_ADDR\_1 → userA → login to → Machine\_1
2. userA → launches a script → "runnable.sh" on Machine\_1
3. From Machine\_1 → userA → Login to → Machine\_2
4. userA → launches a script → "new\_runnable.sh" on Machine\_2
5. userA → changes privilege → becomes 'root'
6. From Machine\_2 → this root (originally - userA) → Log in → Machine\_3
7. 'root' (originally - userA) → executes binary → "collect\_data.sh" on Machine\_3
8. Binary "Collect\_data.sh" as root → internally communicates → to mysql-based service inside datacenter and downloads data
9. root → externally communicates → POSTs the data to external server "EXT\_SERVER\_1"

Here source/entry-point == IP\_ADDR\_1

Data transferred to external server == EXT\_SERVER\_1

Machine transferring data == Machine\_3

User transferring data == root

Actual user hiding behind root == UserA

**Origin user:**

From the above scenario, it can be said the "userA" actually logged in from IP\_ADDR\_1

And sent data to EXT\_SERVER\_1

It is possible that the user changes its privilege more than once but to study the behavior, we always consider the "original user".

**Interactive processes:**

When dealing with user behavior, it is important to collect all the processes that were started with "tty" even though without "ssh".

**Child processes:**

If the user launches a process 'p', it is also important to find all the child-processes that the parent 'p' launched.

**User processes talking to Servers:**

The work done by any process can be attributed to the "user" running the process.

So when we look to investigate breach, we should look at what activity is done by those processes that are launched after the user gets into the datacenter.

So we will also consider all processes or servers (internal and external) that these processes are talking to.

### Machine Class:

We consider machine as a box executing some binaries/servers/processes.

A unique way to classify machines is - to classify them on the basis of binaries/servers/processes they run.

We give an id and a Tag to every group. This group is called Machine Class.

Every machine in the datacenter belongs to a "Machine Class".

Machines belonging to the same "Machine class" will have the same set of binaries/servers running.

Since they run same set of processes, Machines for a given machine-class "M1", will behave be more similar to each compared to a machine of some different Machine class "M2".

Therefore baselining behavior of a Machine based on its Machine-Class helps in finding the anomaly in a datacenter.

### Insider Behavior Graph:

Given the above set of definitions, we implement a GBM model create to graph of connected clusters nodes.

**Each cluster/node in the graph can be either:**

- **a cluster of Users or**
- **a cluster of launched-processes or**
- **a cluster of process-servers running on a machine-class**
- **a cluster External Incoming IpAddress (Clients)**
- **a cluster External Dns (Servers)**
- **a cluster External IpAddress (Servers)**

### Vertical Tiering:

The graph is vertically tiered into

**Tier0:** Source Ip Address/Geoip ->

**Tier1:** (user on machine class) ->

**Tier2:** launched processes and child processes or Interactive processes ->

**Tier3:** (processes/servers+machine class) or ext Dns or ext Ip/GeoLocation

In Tier1 above, we a user login into to a MachineClass is considered as a Node.

In tier2, we group all the processes for a given user and having similar connectivity into the same Node (Similar Connectivity: with respect to the servers they talk and processes they launch)

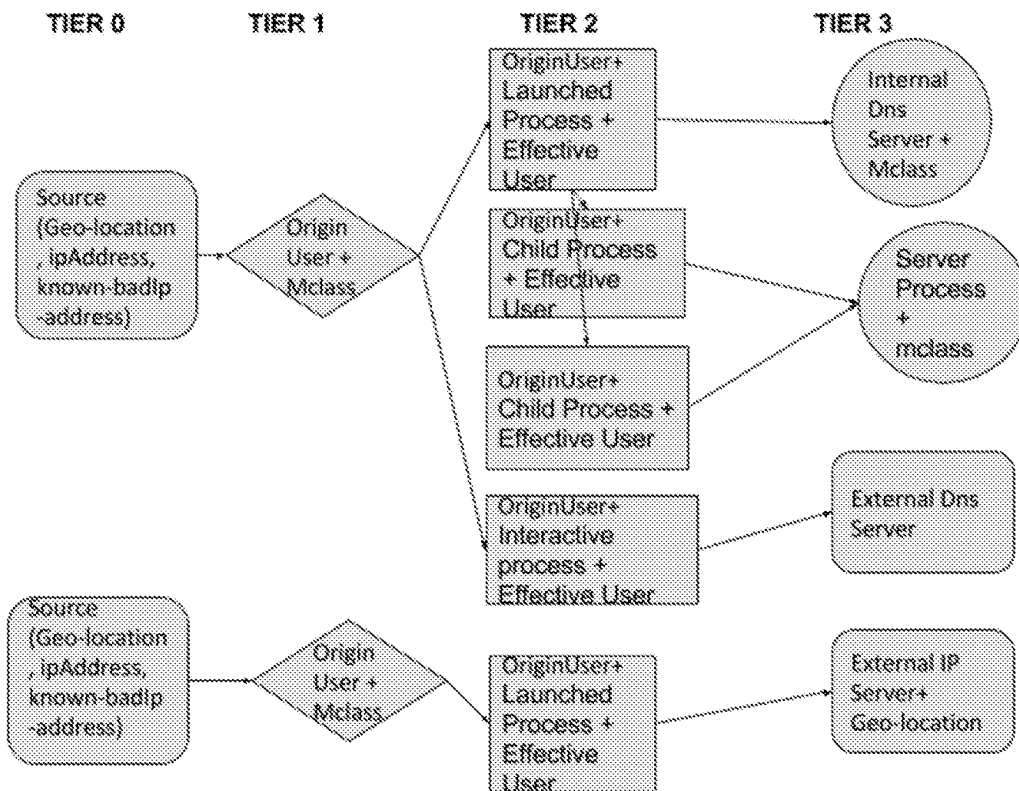
In tier3: These are all the services/servers/Dns/hosts/IpAddress that the the above process talks to.

So the relationships are as follows

Tier0 → LOGIN TO → Tier1

Tier1 → LAUNCHES → Tier2

Tier2 → CONNECTED TO → Tier3



### Horizontal Tiering:

As you see in the above representation, clustering in the graph leads to a very useful side affect  
-> Horizontal tiering in Tier1 and Tier2.

This happens because we are introducing "origin\_user" in Tier1 and Tier2.

This Horizontal tiering is useful because it ensures that there is no Overlap in between any two  
"user" in these two tiers.

### **Edge relationship between Tier1 and Tier2:**

Tier1 represents “user”, say U, logging to a machine, say, which has machine class “M”

Tier2 is a cluster of processes having some ctype C running as effective user “U1” (U1=U or U not=U) and origin user U

Notice that although there is an edge from Tier1 to Tier2, the user in Tier2 may not be the same as the user in Tier1.

The user in Tier1 is the ORIGIN\_USERNAME, which means, this is the real User responsible for launching the process in Tier2.

However, this User may successfully change its privilege and become some other user before launching the same Process.

Our “UserTracking (or Ssh tracking)” approach helps us find out the ORIGIN\_USERNAME for every process.

Also, note, that privilege change of a user from U to U1 take place in the following ways:

1. U becomes U1 on the same machine.
2. U first ssh into a machine. From here, he may keep ssh-ing to other machines.
3. While ssh-ing to different machines U may change privilege to any number of users.

For example U on machine m1 -ssh--> U on machine m2 -becomes-> X on m2 --ssh-> X on m3 -becomes-> U on m3 --becomes-> Y on m3 ..... - launches a process on machine Mn as U1

We hid the above complexity from the user and help the user in directly identifying the effective User in Tier1 and the ORIGIN\_USER behind hiding behind this user.

However, if the User intends to know How User U becomes U1, we do provide the above end-to-end path for that edge.

### **Use of Insider Behavior Graph:**

1. Horizontal Tiering helps in a better visual investigation. In case of a breach or while investigating the User behavior, it becomes faster to search for end-to-end path when we have a non-overlapping path for each ORIGIN\_USER
2. Horizontal tiering also helps us baseline Insider User Behavior.
3. By building this graph every hour, we can detect new/Changes in the EdgeRelationships between the nodes in these tiers. Any change here leads to a change associated with that user.

### Alerts:

Examples of changes we are able to track via this model are:

- User Logs in from a new Ip Address
- User Logs in from an Geo Location not known for that user/datacenter
- User login to a new Machine Class
- User Launching new Process
- User Talking to some internal server that it has never connected to before
- User talking some external server that it has never done before
- User talking to some External Server which is has a geo-location Not known for that user/datacenter

### How does this help Investigation?

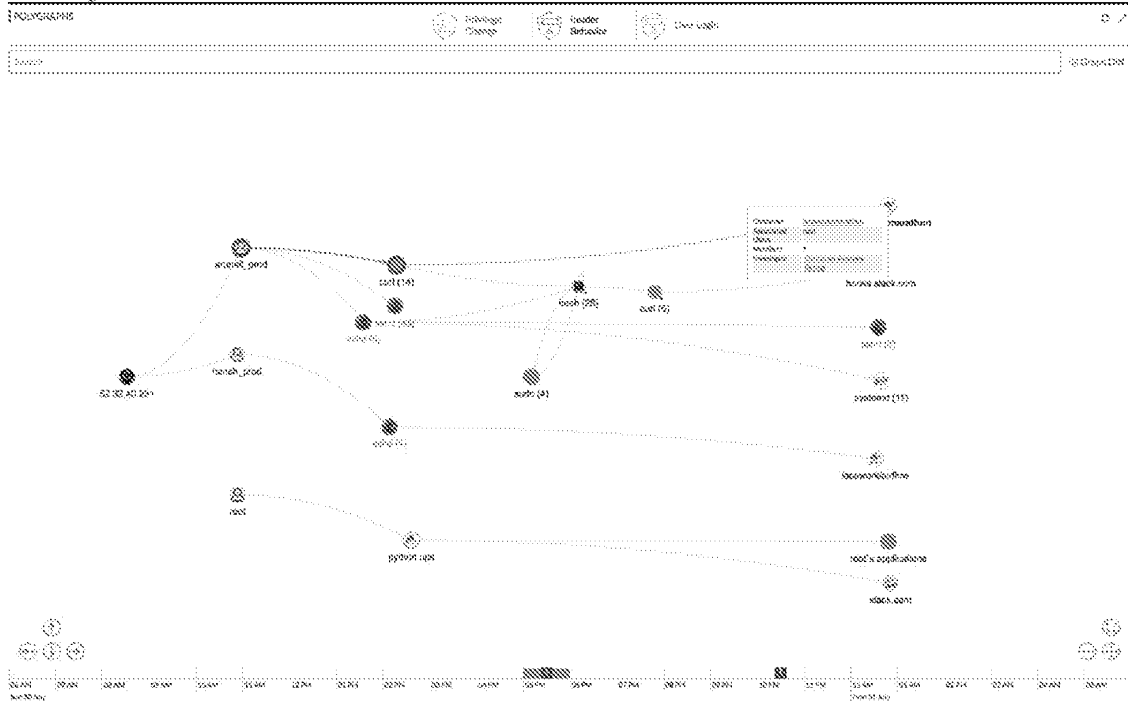
The model will be able to answer any of the following questions a security-ops professional may want to know during a breach:

1. Any new Login Activity? Eg: User tries to login from an Unknown geographic location
2. Suspicious Login Activity: Eg: User tries to login from an Known Bad Ip Address. We identify bad Ip addresses using third party data sources.
3. Usually when a user, coming from outside the datacenter, ssh into a Machine, more often than not, that user will login to a machine of same MachineClass. Eg: in Aws, user always ssh's into a machine called "bastion" first. From bastion, the user can go to any other machine. security best practice, this has to be strictly followed. However, if this is not honored and the user succeeds to directly SSH into some other machine before "bastion", our Insider Graph will catch this and produce an alert.

Also, the graph will give an end-to-end path of this user's activity.

4. User started communicating to some known Bad External Server? We identify bad External Servers using third party data sources.
5. User started communicating to some New External Server in some unknown geographic location?
6. The graph helps us answer questions like - Who is running a particular process?
7. Is root really talking to some external server ? Who is the real user?
8. Has the user logged into a machine and hopped around changing privileges across 500 machine, show the trail each process involved into the trail?

### Example:



Here,

**Tier0:** Source Login ipAddress: 52.32.40.231

**Tier1:** (User + machineClass): aruneli\_prod, harish\_prod, root (machine classes are hidden and not visible in graph)

**Tier2:** Processes that launched by users in Tier1 and their child-processes.

**Tier3:** Internal/External Ips/Servers

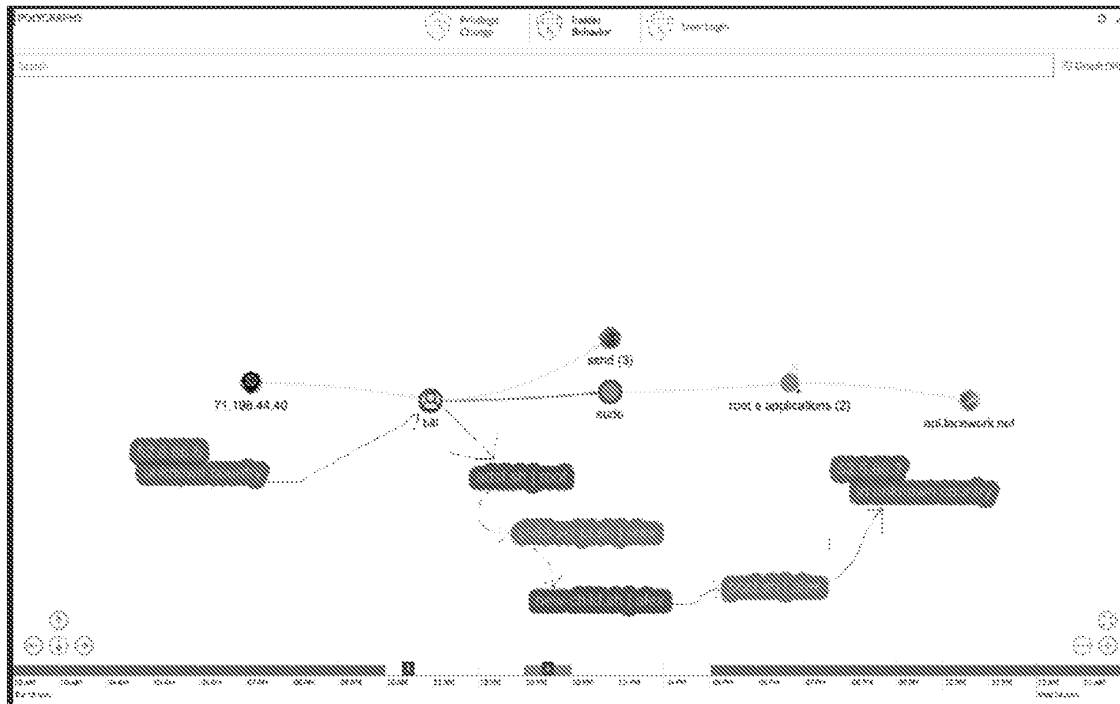
Also, observe the rightmost-top cluster (which says, marathon). This is a representing an Internal Server called “marathon”. Notice the pop-up on that cluster which says the user running this process is “root”.

However if we follow the graph backwards (i.e. right to left), we can figure out that this was actually run by user “aruneli\_prod” who logged-in from IpAddress 52.32.40.231 into a machine which is not the same as the machine on which “marathon” is running.

So although user = “root”, ORIGIN\_USERNAME = “aruneli\_prod”

### Example exfiltration use case:

In the below screen-shot, a test “abnormal behavior” scenario is overlayed on top of “normal behavior” of user “bill”



As you can see, “bill” is a known-everyday user who logs in from some known location and a known Ip address (71.196.44.40).

Usually, “bill” does a “sudo”, becomes root talks to some well-known Dns-es / external servers everyday.

But suppose his credentials gets compromised and somebody is able to get in to the datacenter.

One fine day, this graph detects that some user “alex” inside the datacenter is talking to an external IP 52.5.66.7 - which belongs to a location/country XXX not known before for this datacenter.

Coming to the graph, we start investigation.

We find that “alex” used a script called “post.sh” to talk to 52.5.66.7 which had outgoing-bytes.

Using this graph to further investigate, we find that “post.sh” was launched by a script called “new-child.sh” → which was launched by “child-script.sh”

All this activity was associated to user “alex”

But using ssh-tracking, this graph also knows who the “ORIGINAL\_USER” is.

It turns out the original\_user is “bill”

And “bill” has never run such “scripts”, nor has connected to any External Ip address in that country XXX

On further investigating, we find the source IP address for “bill” which is also some UNKNOWN IPAddress belonging to that country XXX.

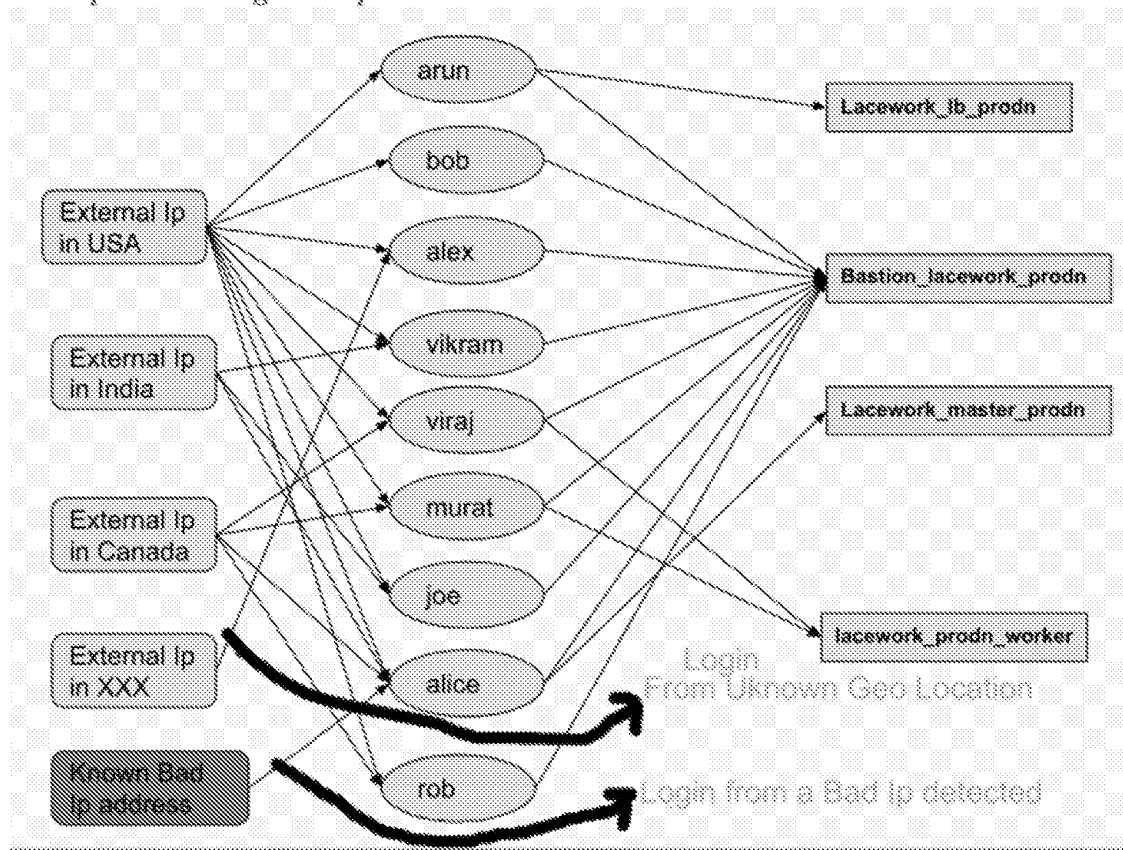
*When will users see this graph?*

As a user of platform 102, users will arrive at the graph when an alert is generated.

In the above scenario, we will be able to combine the following alerts which leads to exfiltration-detection:

1. “Bill” logs in from unknown Geo Location
2. “Bill” launches a new script “adsf.sh”
3. “adsh.sh” launched New child binary “new-child.sh” running “alex” (original\_user - “bill”)
4. “adsf.sh” launched New child binary “post.sh” running as “alex” (original\_user - “bill”)
5. “Bill” connected and sent data to Unknown External Server 52.5.66.7 using “post.sh” running as “alex” (original\_user - “bill”)

Example User-Login Graph:



Tier1: Cluster of Source Ip Address belonging to a specific Geo-location (country)

OR

Known bad ip address

Tier2: User

Tier3: Clusters of “Machine Class”

This graph tracks the normal Login-behavior of the User and provides answers to:

1. Where is the “user” login in from?
2. Which Geolocation/country a login happened from?
3. Did any user come from a known-bad Ip Address?
4. Consider machine-class MC. Has any user accessed a machine belonging to class MC?
5. Has a given user ever logged into to machine of class ?
6. Find all the machine-classes a given user Logs into

### Alerts Generated by User-Login graph:

1. Login From a known-bad Ip address (we maintain a list of known bad ip-addresses using 3rd party data)
2. Login from a NEW Geo-Location (country) for the first time.
3. New User seen in the datacenter for the first time
4. A User logging in to access a machine belonging to MachineClass that he/she has never accesses before. For example:

A user U may normally login to a machine of type “database-servers” but not into “app-servers”. Today if this user logs into some “app-server” which U has never logged in on, this graph will detect it and produce an alert.

### Privilege Change Graph:

A unique way of tracking privilege changes in the datacenter is by monitoring a process-hierarchy of the network-active processes(i.e processes that do network activity. This is important as it filters out a lot of noisy commands/processes like “su -u” )

### Process-Hierarchy:

In unix, each unix process has two ID numbers assigned to it: The Process ID (pid) (of which we create a corresponding PID\_HASH) and the Parent process ID (ppid) .

Each user process in the system has a parent process.

Whilst the operating system can run many processes at the same time, in fact it only ever directly starts one process called the init (short for initial) process. This isn't a particularly special process except that its PID is always 0 and it will always be running.

For example, process-hierarchy may look like this :

```
init—NetworkManager {root}—dhclient {root}
|
|—dnsmasq {root}
|—3*[{NetworkManager}] {root}
|—accounts-daemon {root}—2*[{accounts-daemon}] {root}
|—atd {root}
|—avahi-daemon{avahi}—avahi-daemon{avahi}—padae_run {root}
|—bluetoothd {root}
|—console-kit-dae {root}—63*[{console-kit-dae}] {root}
|—cron {root}
|—cupsd{cupsd}
|—dbus-daemon{dbus}—sshd {root}
|—dd {root}
|—6*[getty] {root}
|—klogd{klogd}—xyz {root}
|—modem-manage {root}
|—mysqld{mysql_user}—16*[{mysqld}]{mysql_user}—dump {root}
|—ntpd {ntpd}
|—packagekitd {root}—2*[{packagekitd}] {root}
|—polkitd {root}—2*[{polkitd}] {root}
|—sshd {root}
|—syslogd {root}
|—systemd-logind {root}
|—systemd-udev {root}
```

```

└─udisksd {root}——4*[{udisksd}] {root}
└─upstart-file-br {root}
└─upstart-socket- {root}
└─upstart-udev-br {root}
└─buddy-friendd {root}——2*[{friendd}] {root}

```

In the above process-tree, I have replaced PID by the effective user running that process.

For example {root} - indicates that the user running that process is “root”

Similarly, {avahi} Indicates that the user running that process is “avahi”.

### Privilege Change:

From the above example, consider this:

```

└─avahi-daemon{avahi}——avahi-daemon{avahi}----padae_run {root}

```

This means user “avahi” became “root” and ran process “padae\_run” whose parent is “avahi-daemon”

### Input to GBM:

From the data that our agent provides us, we create a graph of

Process P1 -- launching ---> a process P2

Both P1 and P2 have corresponding user U1 and U2 associated with it (where U1 = U2 i.e same user or U1 not same as U2)

We use this data as an input to GBM.

### Privilege Change graph:

In GBM, we group all the processes run by a user into the same CLASS (ctype). Based on the process relationships, we get a clustered graph.

### Nodes in the graph:

- Each node in this graph is a cluster of Processes with a “ctype” (also called as “app-type”).
- For a given cluster, all the processes are being executed by one user ONLY.
- Since all the processes in a cluster belongs to the same user, the label used for cluster is “username” .

For example, in the first graph shown below, “dbus”, “kub”, “messagebus”, “root”, “avahi”, “postfix”, “bill”, “Uid=1001” etc are all USERS running all processes which belong to their respective Nodes/clusters.

### Edges:

- An edge from node1 to node2 in the graph would mean “User1” of node1 changed its privilege to “User2” of node2

### Alerts:

1. New User entering the datacenter: Anytime a new user enters the datacenter and runs a process, the graph will show a new node with a new CTYPE/CLASS (as shown in first screenshot below). This would mean a NewUser detected in the datacenter
2. Privilege Change: A edge from Node1 (user1) to Node2 (user2) would mean User1 changed its privilege and became User2.
3. Privilege Escalation: Same as privilege change except here user2 = "root" (see below, screenshot 2) i.e., an edge from Node1 (user1) to Node2 (root)



The screenshot displays the Graph Explorer tool with a graph visualization of legislative data. The x-axis represents time from 1980 to 2020. Nodes represent bills, and edges represent transitions between them. A specific path is highlighted with a red edge, indicating a change in privilege status. Annotations provide context: "There's a red-colored EDGE in the Graph which means 'privilege'" and "Privilege Transition Detection. 'bill' changes privilege and becomes 'root'". A timeline at the bottom indicates a significant event: "D. Davis dies on 06 June 1996".

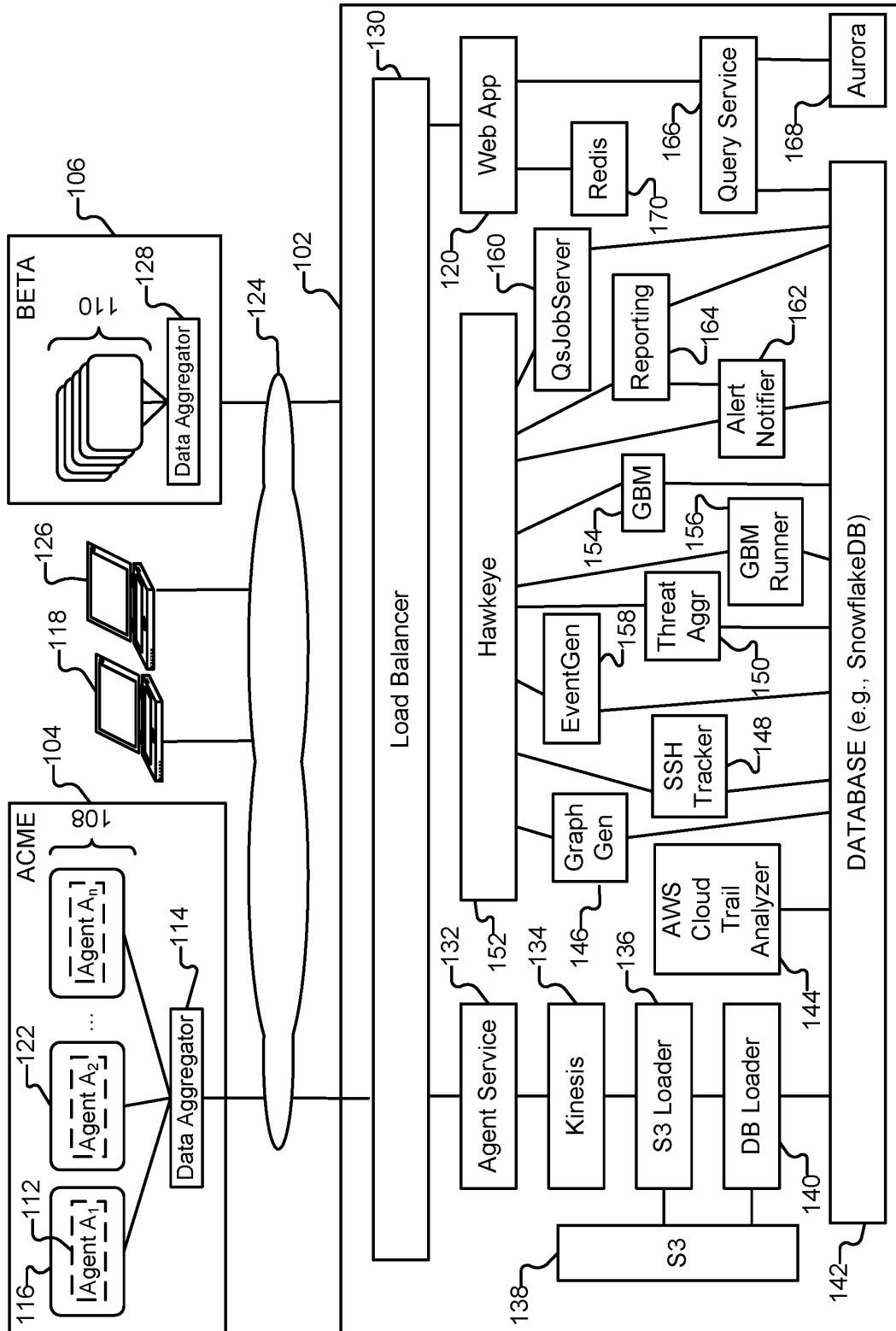


FIG. 1

200 →

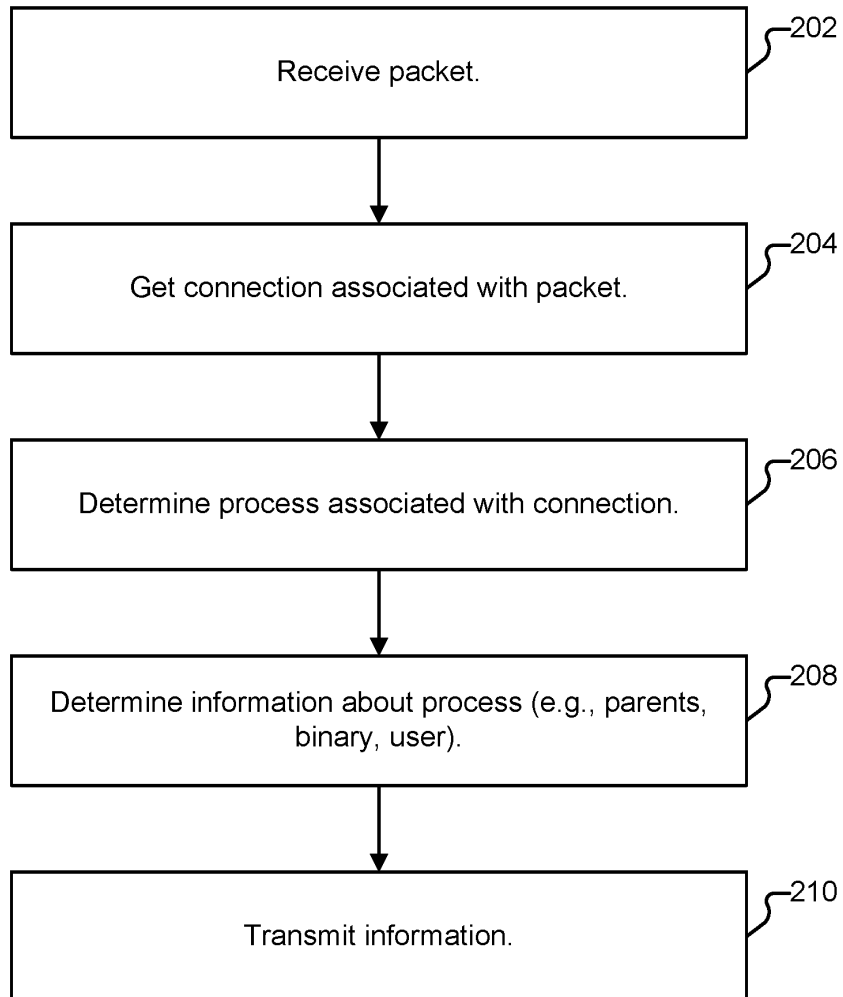


**FIG. 2**

300 →

```
    "event": {
      "created_time": 1501626889179,
      "type": "ProcessData",
      "data": {
306      "net.lacework.model.agent.ProcessData": {
308        "pid": 26191,
310        "pid_hash": 7372148259205580000,
        "start_time": 1501515701990,
        "uid": 0,
        "euid": 0,
328        "username": {
          "string": "root"
        },
        "ppid": 1336,
312        "ppid_hash": {
          "long": 376175681985733950
        },
        "pgid": 26191,
314        "pgid_hash": {
          "long": 7372148259205580000
        },
        "sid": 1336,
316        "sid_hash": {
          "long": 376175681985733950
        },
318      },
320      "tty": "0",
      "cmdline_hash": "1fe756721a20ddec981aa953bce34c1a",
322      "exe_path": {
        "string": "/usr/bin/containerd-shim"
      },
324      "eusername": {
        "string": "root"
      },
326      },
      "container_id": null
    }
  }
}
```

**FIG. 3A**

350 →

```
352  "event": {  
      "created_time": 1501626889179,  
      "type": "ProcessStatsData",  
      "data": {  
354    "net.lacework.model.agent.ProcessStatsData": {  
356      "pid_hash": 7372148259205580000,  
358      "threads": 10,  
360      "vsize": 221245440,  
362      "rsize": 1296,  
364      "utime": 839,  
      "stime": 2852  
    }  
  }  
}
```

**FIG. 3B**

```

"outgoing": {
  "net.lacework.model.agent.UniDirectionData": {
    "bytes": 7524648,
    "compress_ratio": 0,
    "compress_samples": 0,
    "packet_len_hist": {
      "net.lacework.model.agent.Histogram": {
        "count": 471,
        "sum": 7524648,
        "max": 31856,
        "std_dev": 0,
        "buckets": null
      }
    },
    "session_len_hist": {
      "net.lacework.model.agent.Histogram": {
        "count": 4,
        "sum": 7524648,
        "max": 2532151,
        "std_dev": 0,
        "buckets": null
      }
    },
    "session_time_hist": {
      "net.lacework.model.agent.Histogram": {
        "count": 4,
        "sum": 152452,
        "max": 52039,
        "std_dev": 0,
        "buckets": null
      }
    },
    "session_switch_time_hist": {
      "net.lacework.model.agent.Histogram": {
        "count": 3,
        "sum": 544171,
        "max": 223693,
        "std_dev": 0,
        "buckets": null
      }
    }
  }
},

```

**FIG. 3C**

```

"incoming": {
  "net.lacework.model.agent.UniDirectionData": {
    "bytes": 1041,
    "compress_ratio": 0,
    "compress_samples": 0,
    "packet_len_hist": {
      "net.lacework.model.agent.Histogram": {
        "count": 6,
        "sum": 1041,
        "max": 298,
        "std_dev": 0,
        "buckets": null
      }
    },
    "session_len_hist": {
      "net.lacework.model.agent.Histogram": {
        "count": 3,
        "sum": 1041,
        "max": 347,
        "std_dev": 0,
        "buckets": null
      }
    },
    "session_time_hist": {
      "net.lacework.model.agent.Histogram": {
        "count": 3,
        "sum": 87,
        "max": 43,
        "std_dev": 0,
        "buckets": null
      }
    },
    "session_switch_time_hist": {
      "net.lacework.model.agent.Histogram": {
        "count": 3,
        "sum": 1750190,
        "max": 1180754,
        "std_dev": 0,
        "buckets": null
      }
    }
  }
},

```

**FIG. 3D**

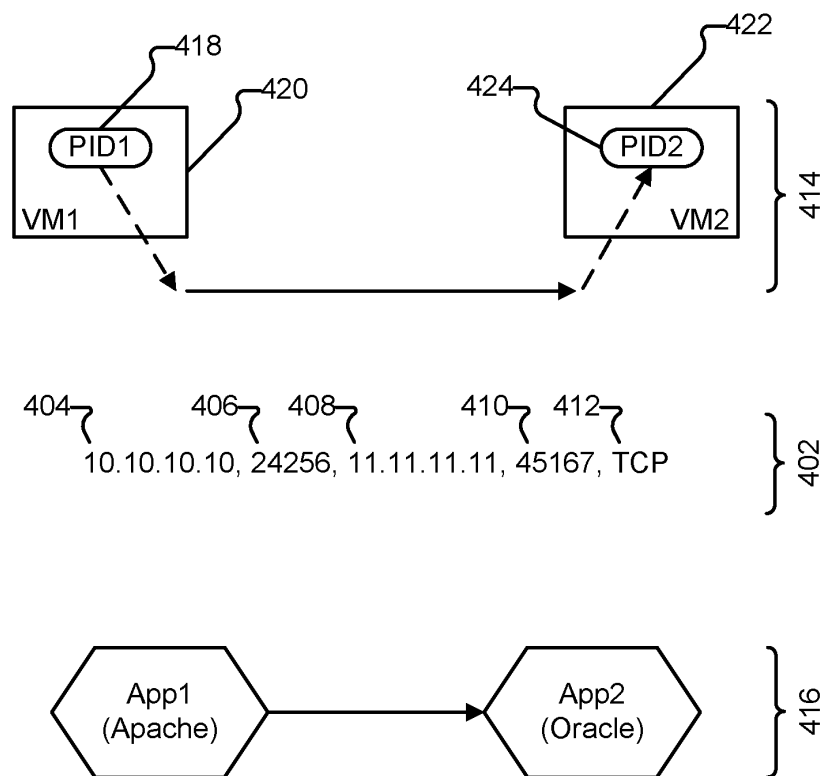


**FIG. 4**

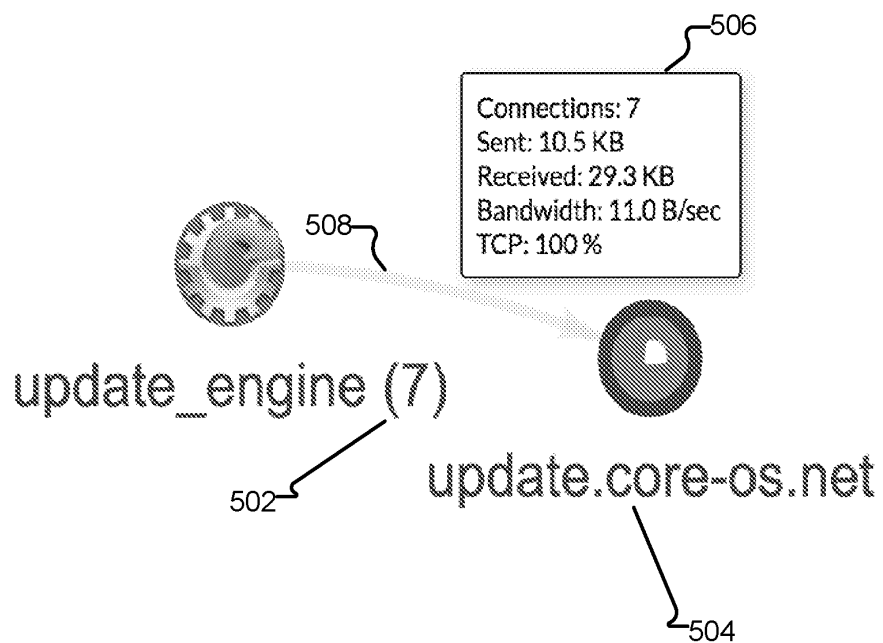


**FIG. 5**

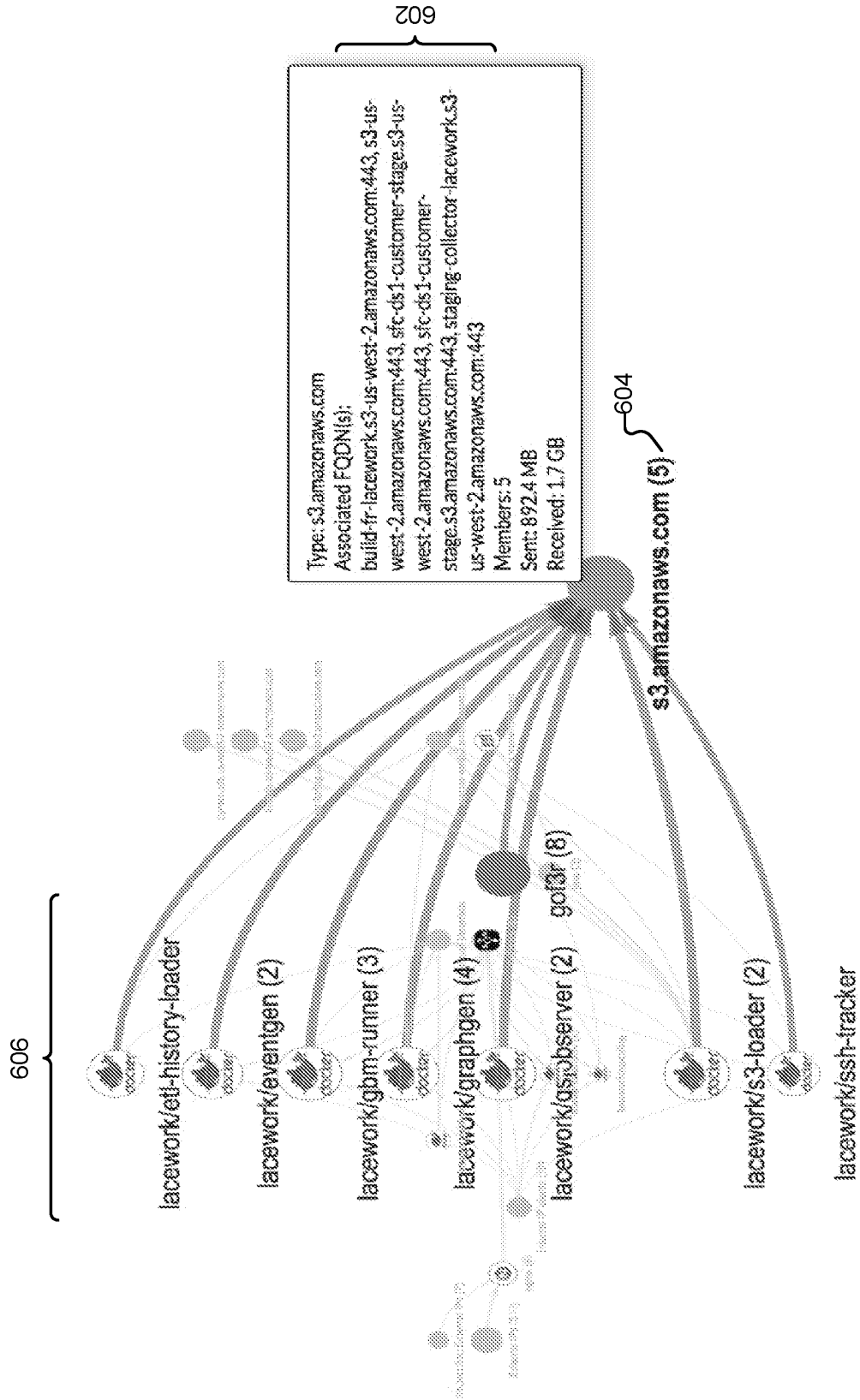


FIG. 6



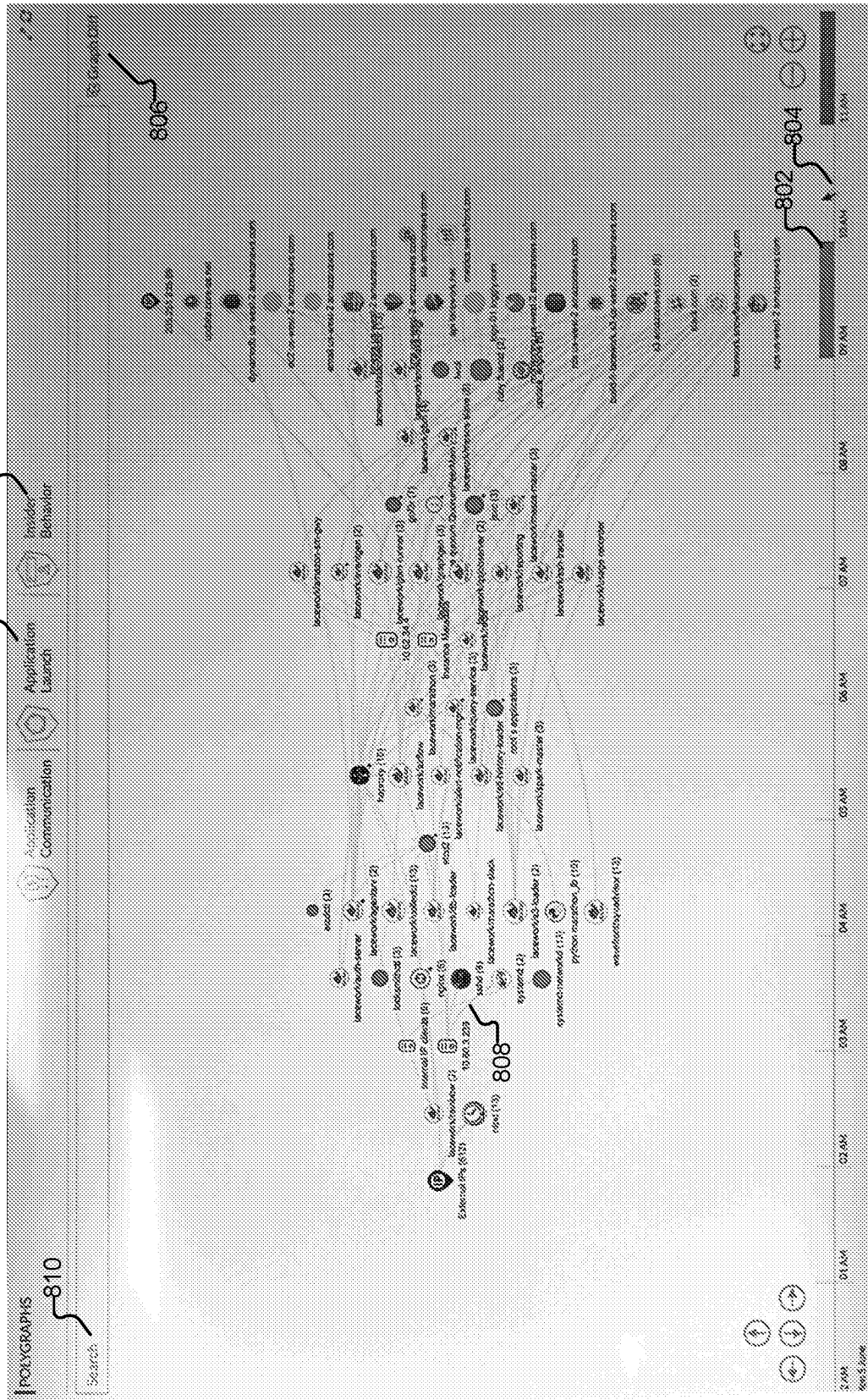


FIG. 8

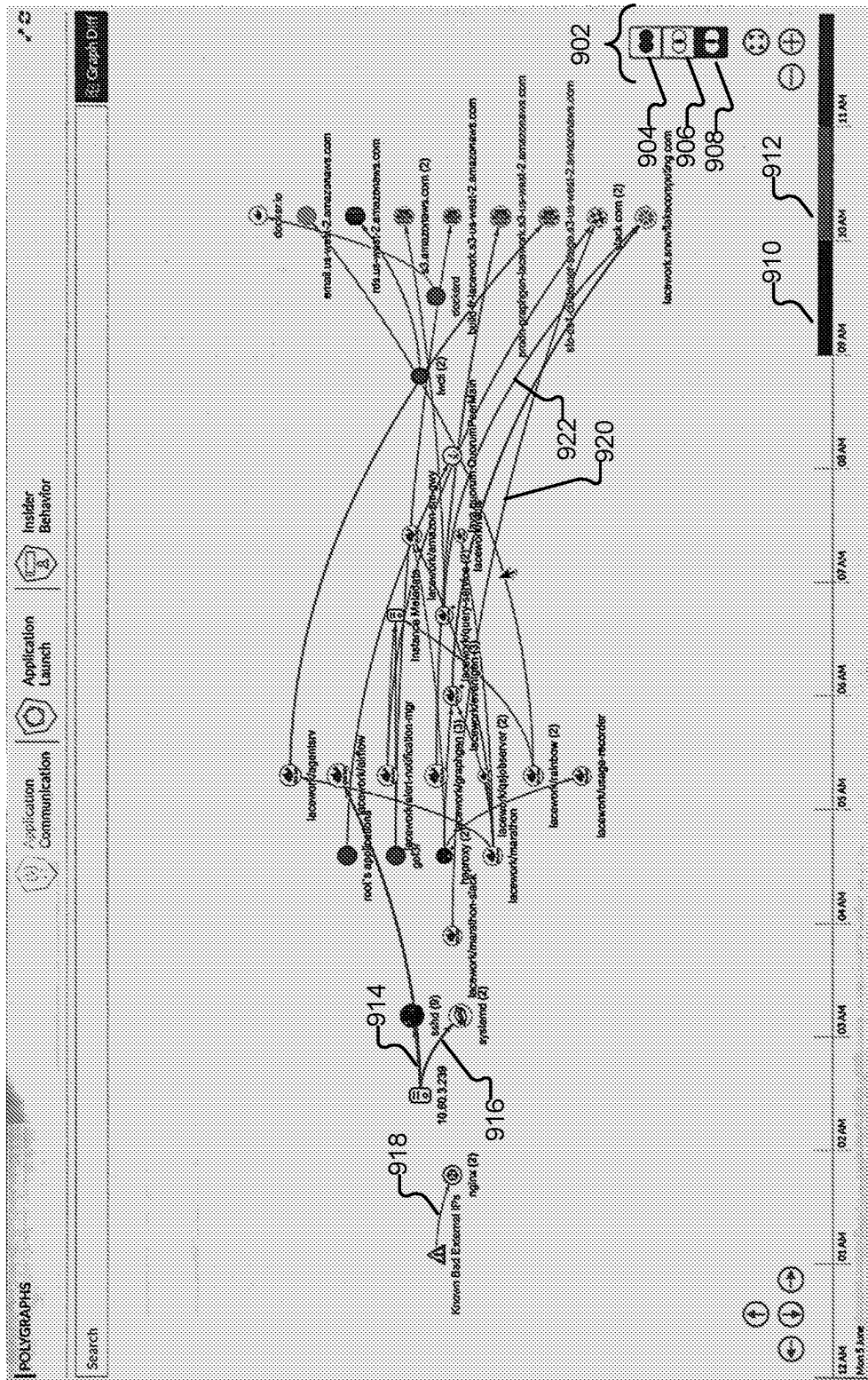
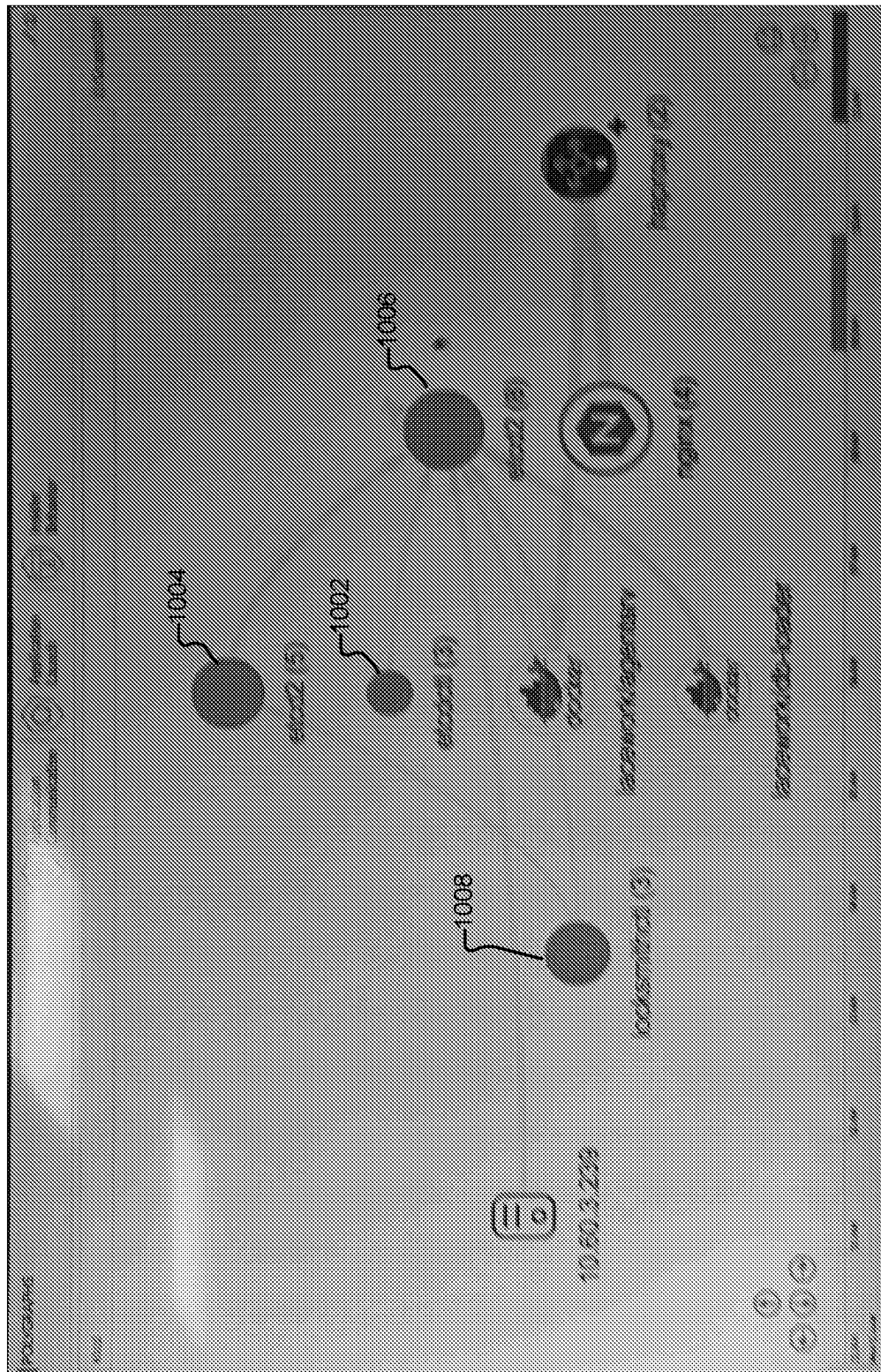


FIG. 9



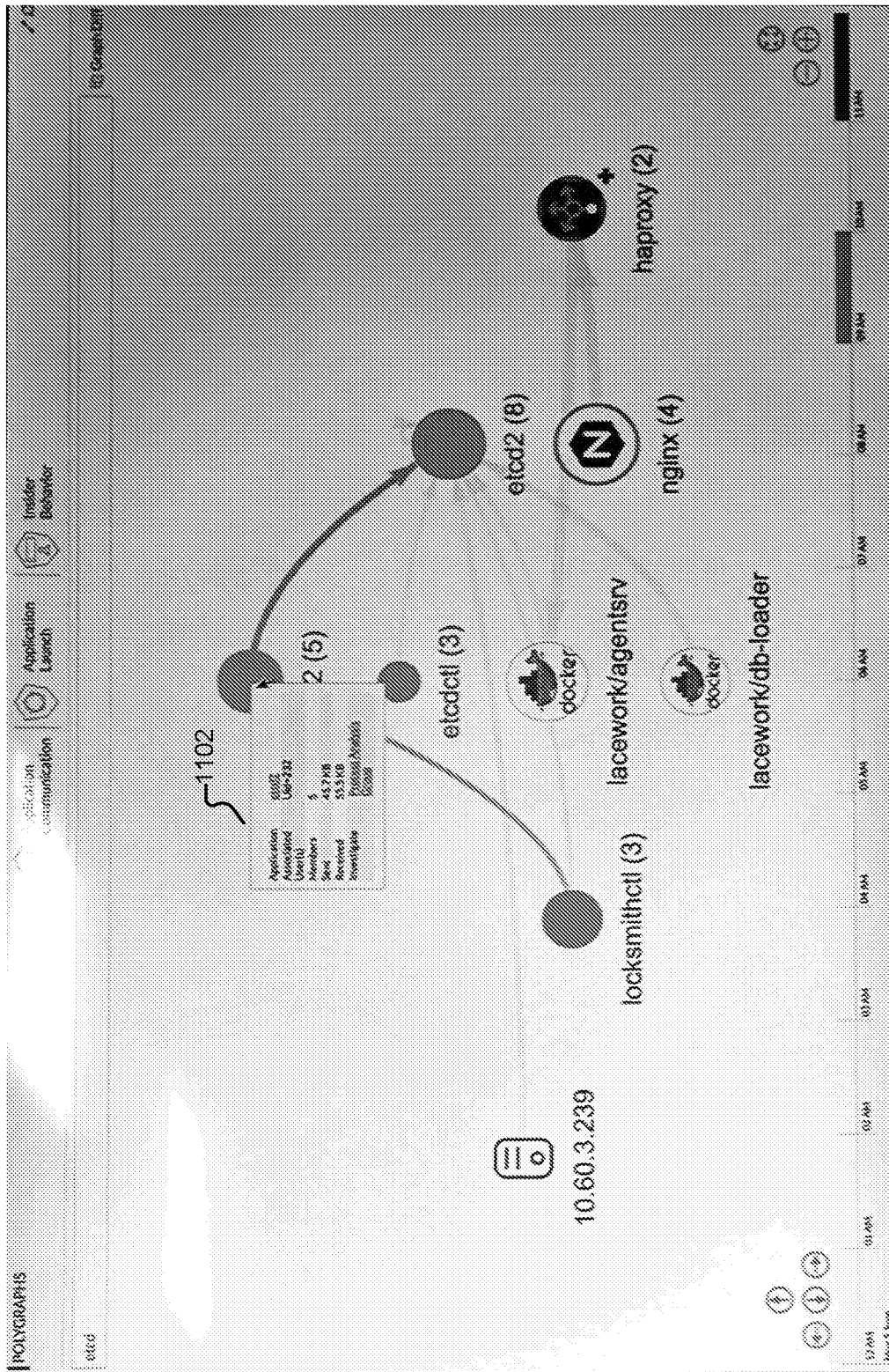


FIG. 11

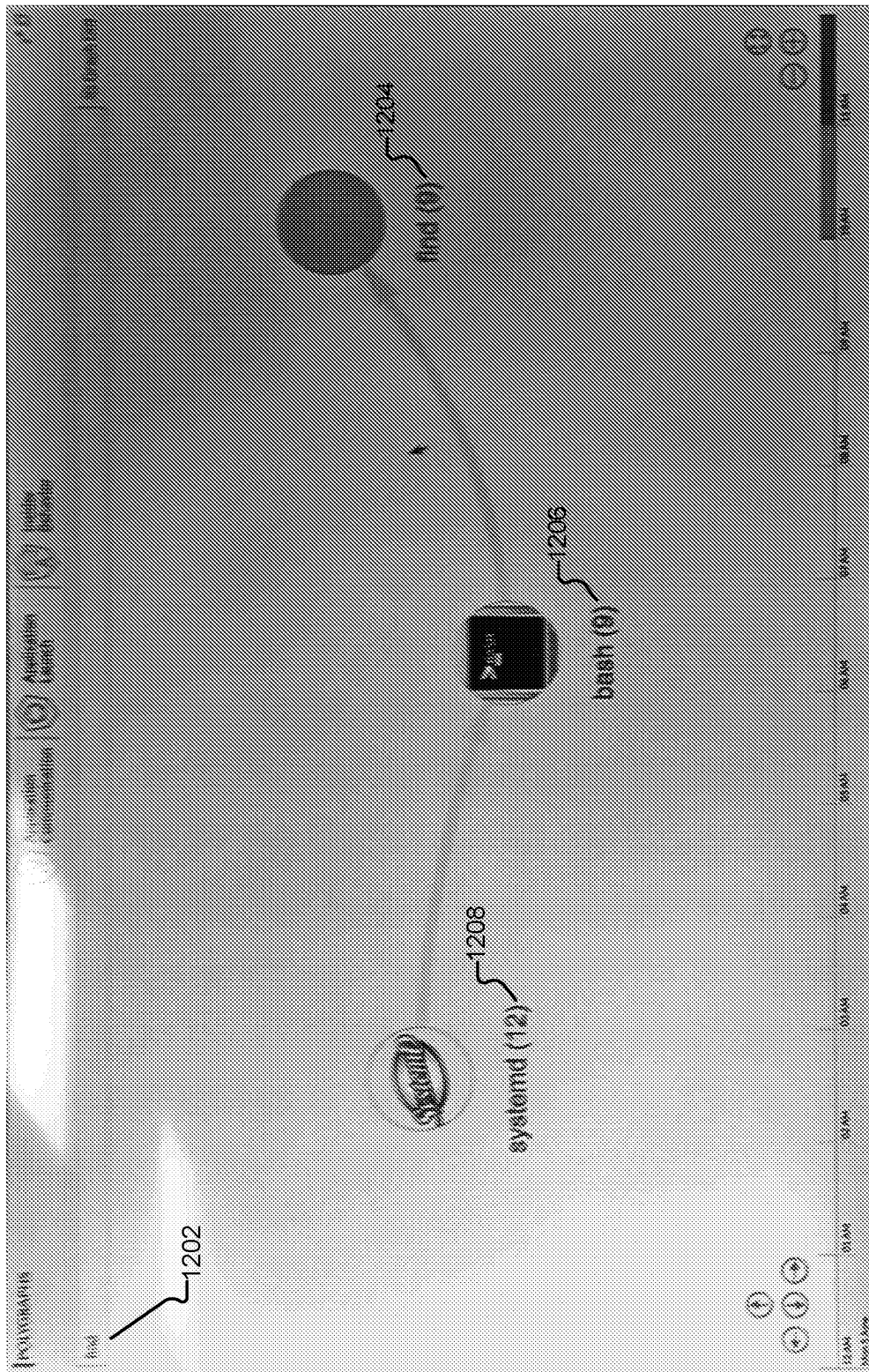


FIG. 12

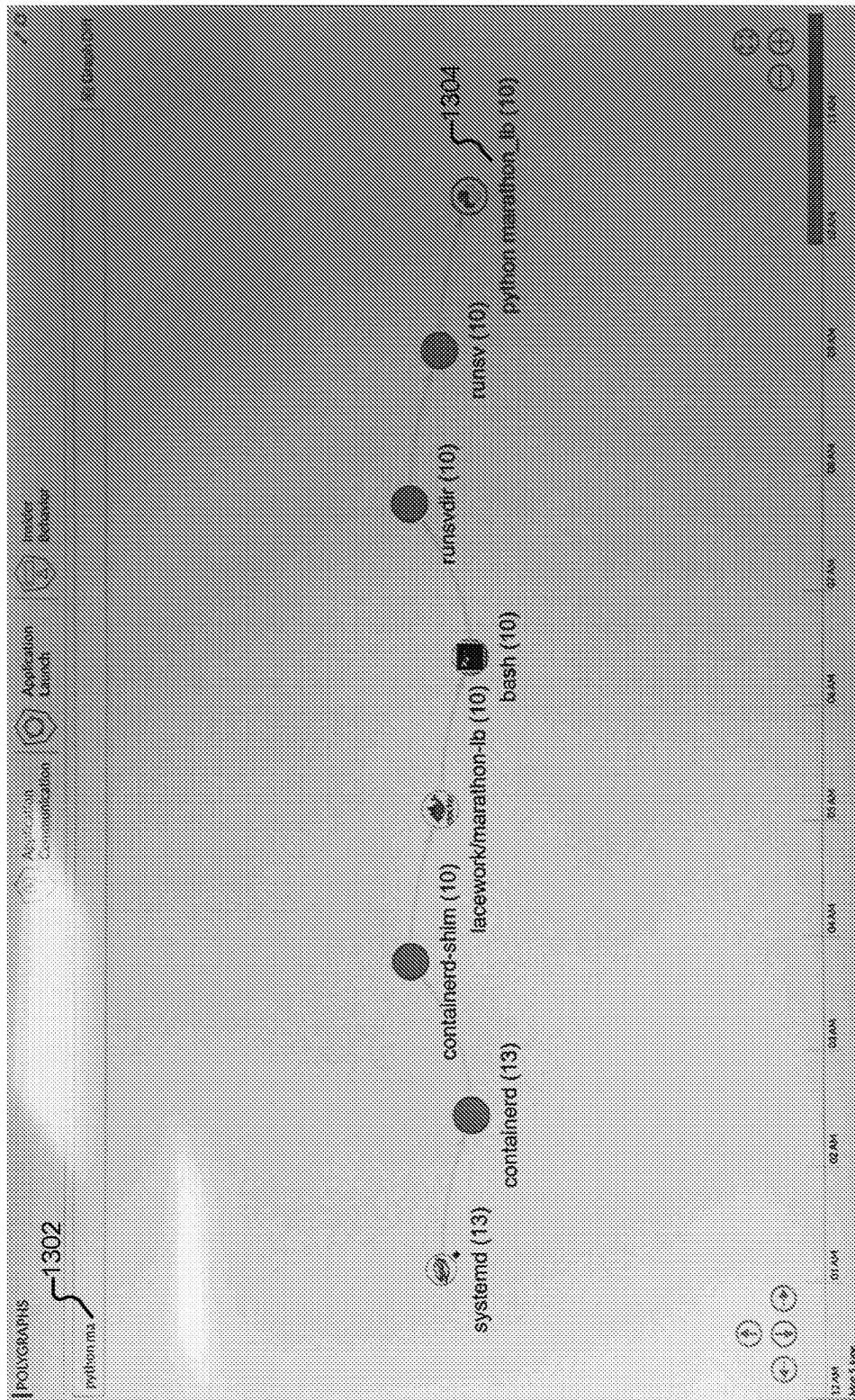


FIG. 13

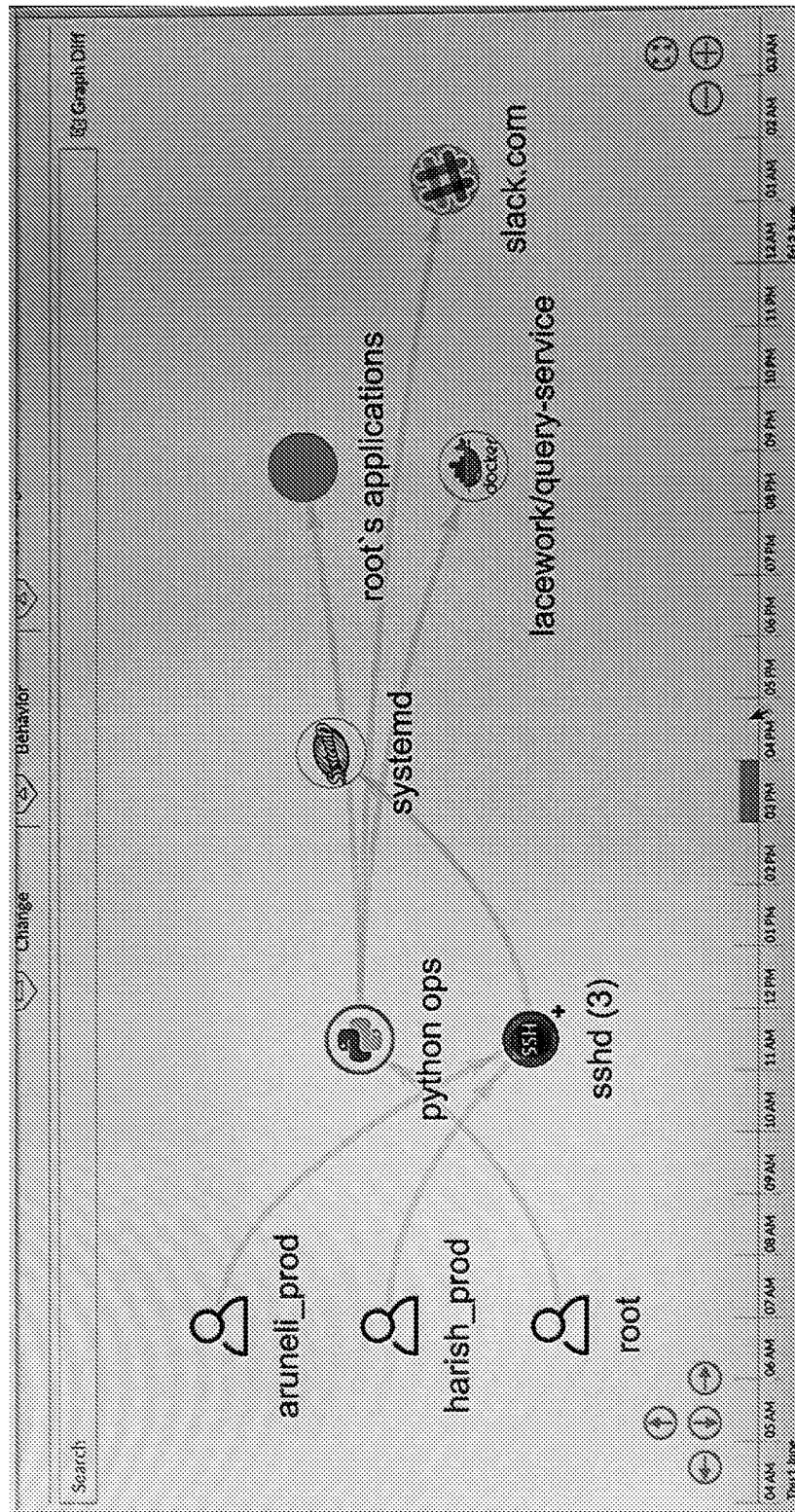


FIG. 14

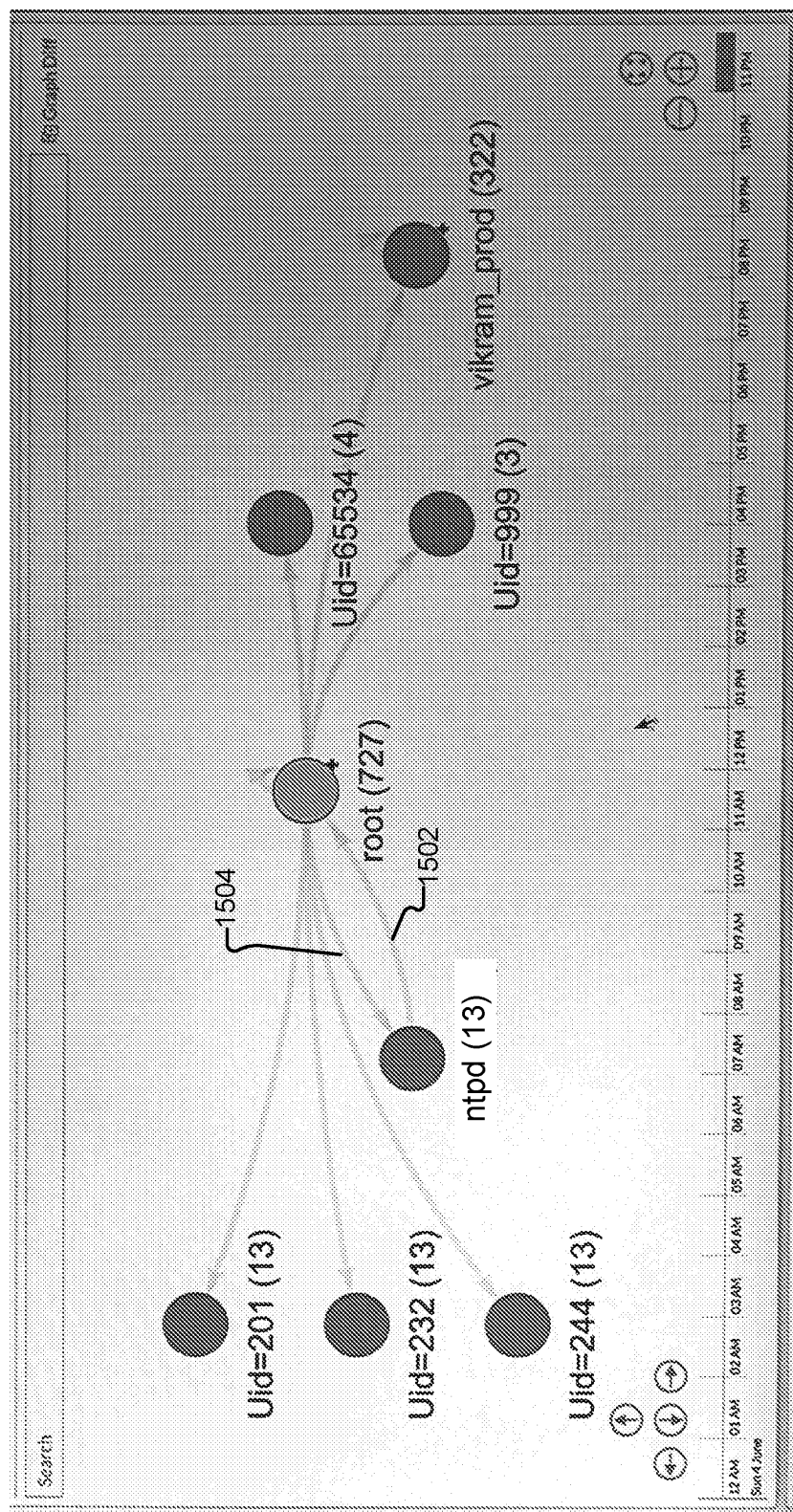


FIG. 15

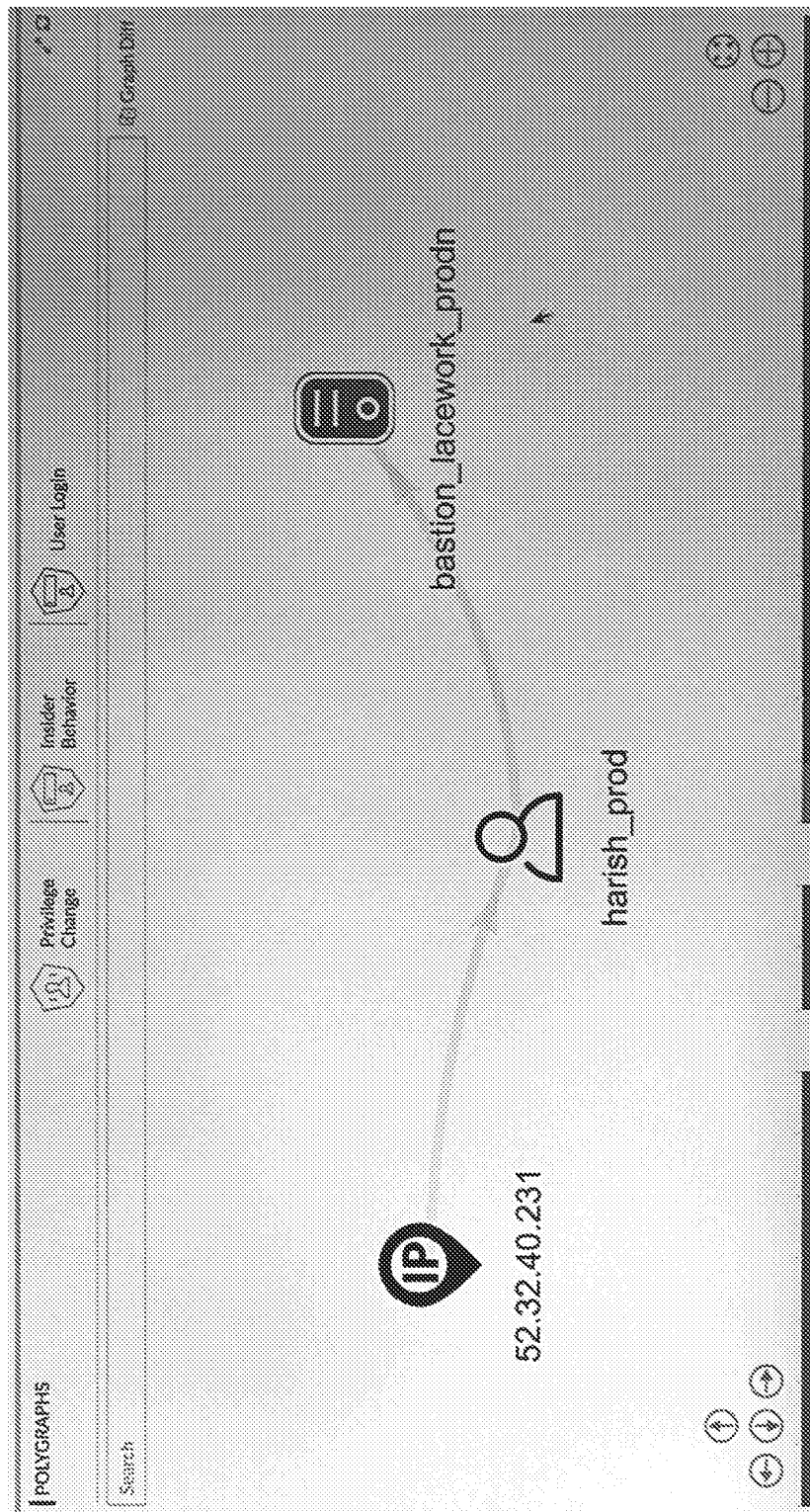


FIG. 16



| Electronic Patent Application Fee Transmittal |          |                                                |        |                      |
|-----------------------------------------------|----------|------------------------------------------------|--------|----------------------|
| Application Number:                           |          |                                                |        |                      |
| Filing Date:                                  |          |                                                |        |                      |
| Title of Invention:                           |          | USING GRAPH-BASED MODELS TO IDENTIFY ANOMALIES |        |                      |
| First Named Inventor/Applicant Name:          |          | Vikram Kapoor                                  |        |                      |
| Filer:                                        |          | Robyn Erinn Wagner/Chelsea Watkins             |        |                      |
| Attorney Docket Number:                       |          | LACEP001+                                      |        |                      |
| Filed as Large Entity                         |          |                                                |        |                      |
| Filing Fees for Provisional                   |          |                                                |        |                      |
| Description                                   | Fee Code | Quantity                                       | Amount | Sub-Total in USD(\$) |
| Basic Filing:                                 |          |                                                |        |                      |
| PROVISIONAL APPLICATION FILING                | 1005     | 1                                              | 260    | 260                  |
| Pages:                                        |          |                                                |        |                      |
| Claims:                                       |          |                                                |        |                      |
| Miscellaneous-Filing:                         |          |                                                |        |                      |
| Petition:                                     |          |                                                |        |                      |
| Patent-Appeals-and-Interference:              |          |                                                |        |                      |
| Post-Allowance-and-Post-Issuance:             |          |                                                |        |                      |

| Description               | Fee Code | Quantity | Amount | Sub-Total in USD(\$) |
|---------------------------|----------|----------|--------|----------------------|
| <b>Extension-of-Time:</b> |          |          |        |                      |
| <b>Miscellaneous:</b>     |          |          |        |                      |
| <b>Total in USD (\$)</b>  |          |          |        | <b>260</b>           |

| Electronic Acknowledgement Receipt          |                                                |
|---------------------------------------------|------------------------------------------------|
| <b>EFS ID:</b>                              | 31051418                                       |
| <b>Application Number:</b>                  | 62590986                                       |
| <b>International Application Number:</b>    |                                                |
| <b>Confirmation Number:</b>                 | 1075                                           |
| <b>Title of Invention:</b>                  | USING GRAPH-BASED MODELS TO IDENTIFY ANOMALIES |
| <b>First Named Inventor/Applicant Name:</b> | Vikram Kapoor                                  |
| <b>Customer Number:</b>                     | 21912                                          |
| <b>Filer:</b>                               | Robyn Erinn Wagner/Chelsea Watkins             |
| <b>Filer Authorized By:</b>                 | Robyn Erinn Wagner                             |
| <b>Attorney Docket Number:</b>              | LACEP001+                                      |
| <b>Receipt Date:</b>                        | 27-NOV-2017                                    |
| <b>Filing Date:</b>                         |                                                |
| <b>Time Stamp:</b>                          | 17:31:08                                       |
| <b>Application Type:</b>                    | Provisional                                    |

### Payment information:

|                                                                                                                                                                                                                                                                                |                             |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|
| Submitted with Payment                                                                                                                                                                                                                                                         | yes                         |
| Payment Type                                                                                                                                                                                                                                                                   | DA                          |
| Payment was successfully received in RAM                                                                                                                                                                                                                                       | \$260                       |
| RAM confirmation Number                                                                                                                                                                                                                                                        | 112817INTEFSW00009206500685 |
| Deposit Account                                                                                                                                                                                                                                                                | 500685                      |
| Authorized User                                                                                                                                                                                                                                                                | Chelsea Watkins             |
| <p>The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:</p> <p>37 CFR 1.16 (National application filing, search, and examination fees)</p> <p>37 CFR 1.17 (Patent application and reexamination processing fees)</p> |                             |

37 CFR 1.19 (Document supply fees)  
37 CFR 1.20 (Post Issuance fees)  
37 CFR 1.21 (Miscellaneous fees and charges)

### File Listing:

| Document Number              | Document Description                        | File Name                    | File Size(Bytes)/<br>Message Digest      | Multi Part /.zip | Pages (if appl.) |
|------------------------------|---------------------------------------------|------------------------------|------------------------------------------|------------------|------------------|
| 1                            | Application Data Sheet                      | LACEP001Plus_ADS.pdf         | 1823084                                  | no               | 10               |
|                              |                                             |                              | 862bed90c08e116e289eb8c8c6e6561c766c7bec |                  |                  |
| Warnings:                    |                                             |                              |                                          |                  |                  |
| Information:                 |                                             |                              |                                          |                  |                  |
| 2                            | Specification                               | LACEP001Plus_APP.pdf         | 3539005                                  | no               | 104              |
|                              |                                             |                              | 2d7bfc173f895fba89fda171e1f6f6a9247bc0c5 |                  |                  |
| Warnings:                    |                                             |                              |                                          |                  |                  |
| Information:                 |                                             |                              |                                          |                  |                  |
| 3                            | Drawings-only black and white line drawings | LACEP001Plus_APP_Figures.pdf | 7390515                                  | no               | 19               |
|                              |                                             |                              | c13d01d887a97cd3df1c7f3835cd317a3381ed54 |                  |                  |
| Warnings:                    |                                             |                              |                                          |                  |                  |
| Information:                 |                                             |                              |                                          |                  |                  |
| 4                            | Fee Worksheet (SB06)                        | fee-info.pdf                 | 30023                                    | no               | 2                |
|                              |                                             |                              | df5ac0e33b25fc18e2a95a462d16104bf2ca6ba7 |                  |                  |
| Warnings:                    |                                             |                              |                                          |                  |                  |
| Information:                 |                                             |                              |                                          |                  |                  |
| Total Files Size (in bytes): |                                             |                              | 12782627                                 |                  |                  |

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

**New Applications Under 35 U.S.C. 111**

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

**National Stage of an International Application under 35 U.S.C. 371**

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

**New International Application Filed with the USPTO as a Receiving Office**

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.