Data Analytics for Incident Response

Samir Saklikar & Dennis Moreau
Office of the CTO, RSA, The Security Division of EMC
Critical Incident Response
Organized Approach to addressing and managing the aftermath of a security breach or attack

- Detection & Correlation (AT, 0-Day Scenario)
- Limit Damage
- Reduce Recovery time
- Learn (to) and Adapt (from) the attack
- Protect but Monitor
- Collect Data to Litigate
Aurora-like attacks –

1. Infection Steps
   a. Social Engineering to visit malicious website
   b. User Browser Vulnerability to load custom malware
   c. Malware contacts C&C
   d. Privilege Escalation on local network
   e. Dump Active Directory and remotely crack credentials
   f. Gain VPN access

2. Stealing steps
   a. Highly Targeted for the enterprise or even department!
   b. Human-Controlled (The services of Bots are no longer required)
Ok! But, is it really a hard problem*?

- Infection point may be way in the past
  - Missing Logs, Newer Configuration..
- Distributed in time
  - Gap of days or even weeks between steps
- Distributed in space
  - Use different endpoints for different steps
- Identifying a single step in attack is not enough
  - Need the Attack vector to identify the next step

*hint = Answer is “Yes”
Harder than finding Needles in Haystacks?

- Hmm. Yes!
- Actually, more like finding a blade of grass in a haystack
  - That blade of grass grew in A field, was of B height, C width, D green-ness and was cut at E hours on F date

Find in
Intrusion Detection
  – Host and Network-based tools
SIEM/Packet Capture Tools
Vulnerability Scanners
Memory Analysis
  – WinDD, MDD, Volatility
Disk Analysis
Incident Response – The Data

- Raw Logs, Packet Capture, NetFlow
- Asset Configuration/Vulnerability/Criticality
- Regulatory Controls/Security Architecture
- Topology
- Events, Alerts
- Attacks
- Threats
- Disk/Memory Images
Incident Response – The Challenge!
Big Data
Hmm.. Various Definitions!

451 Group

“Big data is a term applied to data sets that are large, complex and dynamic (or a combination thereof) and for which there is a requirement to capture, manage and process the data set in its entirety, such that it is not possible to process the data using traditional software tools and analytic techniques within tolerable time frames.”

OK! You get the idea.
Big Data Analytics (ref – Wikibon)

Abstraction Layers
- Analytic Applications
- Fast-loading Analytic Database
- Modeling
- Management & Security
- Higher Level Languages
- Job & Task Trackers
- Location-aware File Systems
- Processing & Original Data

ETL (Extract, Transform Load) Modeling Tools, e.g., CR-X

ClickFox, Merced, Etc.

Greenplum
Netezza

Other Data Sources

Hadoop

Cascading

Kerberos

Pig

Hive (DW)

MapReduce Engine

File System, e.g., HDFS

NoSQL DB e.g., Hbase Cassandra

SECURITYBYTE 2011
Map Reduce (Programming Model)

Map: (in_key, in_value) → a list of (mid_key, mid_value)
Reduce: (mid_key, a list of mid_value) → a list of (out_key, out_value)

Data Flow:

Map:
- File1: a b b
- File2: c b c

Reduce:
- Mid_key: a b c
- Mid_value: 1 1 1

GroupBy:
- Mid_key: a b c
- Mid_value: 1 1 1

Reduce:
- Out_key: a b c
- Out_value: 1 3 2
Map Reduce (System)
Scalability in Big Data

- **Mahout**
  - Scalable Machine Learning over Hadoop
  - Clustering, Classification, Naïve Bayes etc

- **Madlib over SQL/Postgres**
  - Data Parallel implementation over SQL
  - Supervised & Unsupervised Learning

- **R interface with SQL**
Evolving IT landscape

- More Assets in the Infrastructure to be managed
  - Move towards cloud; large interdependent assets
- More layers in the Technology stack to monitor
  - Virtualization. More Layers → More Logs
- More Detailed Context required
  - Advanced Threats
- More Security Data Sources
  - Netflow, FPC, Sandbox Indicators
Multiple massive aggregations of loosely structured data

- Large = Logs for all endpoints in enterprise/cloud
  - Logs across all stacks (HW, VMM, OS, APP, Service, …)
- Distributed = Multiple sensors
- Loosely Structured = Log formats (developer-defined strings), Packet Captures
- Multiple Log consumers => multiple analytics and representations (Analysts, Auditors, Ops Problem Resolution, Optimization …)
Big Data for Incident Response - Why

- **Analytics**
  - Tractability over large datasets
    - A Query failure is NOT acceptable
  - Heterogeneity
    - Handle rich data sources
  - Iterative
    - IR team needs to try out various search options

- **Predictable Response times**
  - Limit Damage, Limit Dwell Time
A Typical Incident Response System

- From ‘Evolution of Incident Response’ Blackhat 2004
  - Pre-Incident Preparation
  - Detection of Incidents
  - Initial Response
  - Formulate Response Strategy
  - Data Collection
  - Data Analysis
  - Reporting
Detection of Incidents
Challenges – Manageable Alerts

- Alerts → Incidents?
- Alerts → Viewed Alerts → Incidents
- Alerts v/s CIRT capacity?
  - High Visibility, High Frequency – Handled
    - Alerts >> CIRT capacity
  - Low Visibility, Low Frequency – Ignored
    - Alerts >>>>> CIRT capacity
Dealing with large number of High Visibility Alarms

- Alert Normalization
- Alert Classification
- Alert Clustering
- High Level Alerts
High Confidence in Alert Classification from IDS?

(Answer depends on maturity of IDS rules, IR processes)

- Map Reduce over a certain time-interval of Alert Data
  - Time Interval granularity decided by the Alert Types
- Map: \{in\_key = log-id, in\_value = log-data\} →
  - \{mid\_key = AlertType, mid\_value = transformed-log-data\}
- GroupBy over the AlertType
- Reduce: \{mid\_key = AlertType, mid\_value = Alert Data\} →
  - \{High Level Alerts\}
Reduce Implementation - K-Means Alert Clustering

(An Unsupervised Learning technique)

Alerts need to be represented in a n-dimensional vector space, where n is the number of identifying attributes of the Alert

A Distance measure should be definable over the n-dimensions

- Time
- Distance between IP Addresses?
  - Sub Network Distance
  - Reverse DNS Lookup + Inputs from Fast-Flux Analysis?
  - Reverse DNS + Registration Details
- Destination Port Numbers
- Create n-dimensional numerical representation vector

- Leverage Mahout or Madlib over the Data to cluster into High-Level Alerts
Reduce Implementation – Decision Tree

(A Supervised Learning technique)

A tree-structure with ‘decision nodes’ containing test attributes, branches as possible attribute values, and leaf nodes as classification answers

Data features come from a discrete set of variables
- No Need for a Distance measurement function
- Well suited for some types such as IP, Port
- Time ? – May be converted into quantum units of difference from a certain epoch for the dataset

- Leverage Mahout (Random Forests) algorithms
- Or use Madlib support for Decision Tree learning
Low Confidence in Alert Classification from IDS?

- **Alert Classification**
  - Needs Supervised Learning
  - Decision Tree Learning seems natural fit
  - Decision Attributes – IDS Alert Type, Alert Attributes
  - Answer - AlertType

- **Alert Clustering**
  - K-means Clustering
Current Handling?
  – Mostly Ignore
  – Seen only when something goes wrong

But,
  – This is where APT behavior lurks
  – Distributed in time, Distributed in Space

So,
  – Alerts Scavenging – Very Very Low Success

Hence,
  – Leverage Parallel execution for high-speed trial-and-error searchers
Data Analytics for Low Frequency, Low Visibility Alerts?

- Multi-Level Clustering?
  - 1st Pass on IP & varying time-intervals
    • Identify cluster of Alerts, related to same IP
    • May need multiple iterations with different time-intervals over the dataset
  - 2nd Pass on Alert Types
    • Needs Input on “closeness” of Alert Types, based on which Alert types may follow another
  - Or the other way round? Repeat.
  - Desired Output
    • A cluster of Alerts, related to same IP address within certain time-intervals, wherein Alerts seem to convey a pattern
Challenges – Any Incident Detection (however fast) will be still slow

- **IR team solves Targeted Incidents**
  - **Zeus Family**
    - Highly differentiated behavior within the family
    - Low commonality to write rules
      - Generic HTTP GET Requests
      - Content-Type: text/html
    - Rules based on domain-names
    - Yet, “suspicious” DNS lookup behavior
      - Multiple DNS lookups to all listed DNS servers
      - Seemingly random intervals
      - Lookup successful but no configuration file found
  - Learning stays with IR team i.e. within the human realm
  - Ways to re-use Learning for Faster Incident Detection?
Requirements
- Primary Inputs – Alerts
- Secondary Inputs – Configuration, Patches, CVE Data
- External Inputs – “World context”, “CIRT thoughts”
- Output ? – Close, Categorize, Escalate

Naïve Bayes Classification
- Stocashtic Model wherein Input ‘Independent’ Variables contribute ‘Independently’ towards probability of a data belonging to class C
- ‘Independent’ – makes it manageable;
Naive Bayes Implementation

- Mahout & Madlib support it
- Sample Program over Madlib (yes, that’s it)

```
sql> SELECT madlib.create_nb_prepared_data_tables(
    'training-table','class-col','attributes-col',num-attr, 'nb_feature_probs', 'nb_class_priors');

sql> SELECT madlib.create_nb_classify_view (
    'nb_feature_probs', 'nb_class_priors','to-classify-table', 'id-col', 'attributes-col', num-attr, 'output-class-table');
```
A Typical Incident Response system

- From ‘Evolution of Incident Response’ Blackhat 2004
  - Pre-Incident Preparation
  - Detection of Incidents
  - Initial Response
  - Formulate Response Strategy
  - Data Collection
  - Data Analysis
  - Reporting
Initial Response
Initial Response

- Rubber hits the Road!
  - Limit Damage
  - Reduce Recovery time

- More time the better to...
  - Learn (to) and Adapt (from) the attack
  - Protect but Monitor
  - Collect Data to Litigate
Initial Response – How should it work?

- **From Input**
  - Information about a possible incident
    - a leaked document found on underground sites
    - strong evidence of a compromise

- **To Output**
  - A small set of potential suspected sources of infection on listed endpoints
Step 1 – Finding all accesses to leaked document versions

- Identify all endpoints which have accessed the leaked document
  - Document Server logs
  - Email Logs

- Log Transformation/Normalization
  - Map Reduce Implementation
  - Map: \{\text{in\_key} = \text{log\_file}, \text{in\_value} = \text{log\_data\_id}\} \rightarrow
    - \{\text{mid\_key} = \text{document\_version\_id}, \text{mid\_value} = \text{transform\_log\_data\_id}\}
  - GroupBy = \text{document\_version\_id}
  - Reduce: \{\text{mid\_key} = \text{document\_version\_id}, \text{mid\_value}\} \rightarrow
    - \{\text{out\_key} = \text{document\_version\_id}, \text{out\_value} = \text{Endpoint Ids who accessed document\_version}\}
Reasoning
– If a document was lost from a particular endpoint,
  • directly or via an internal staging site
– the malware may have triggered some alerts from the endpoint

Alert Normalization & Endpoint Correlation
– Map: {in_key = alert-data-file, in_value = alert-data-id} ->
  • {mid_key = endpoint-id, mid_values = transform-alert-data-id}
– GroupBy = endpoint-id
– Reduce: {mid-key = endpoint-id, mid_values} ->
  • {out_key = endpoint-id, out_value = List of un-resolved Alerts at that Endpoint}
Reasoning
– Un-resolved Alerts per Endpoint, will vary through time. Need to focus on a manageable time-period.

Alert Partitioning into Time Periods
– Identify time-periods around the document access time for that Endpoint
– Map: \{in\_key = alert-data-file, in\_value = alert-data-id\} ->
  • \{mid\_key = time-period-id, mid\_values = alert-data-ids\}
– GroupBy = time-period-id
– Reduce: \{mid\_key = time-period-id, mid\_values\} ->
  • \{out\_key = time-period-id, out\_value = List of un-resolved Alerts at that Endpoint within that time period\}
Step 3 - Correlate Un-resolved Alerts with any Server Access from Endpoints

- Within time-window of each of the Alerts, identify all Server access to the endpoints

- Identify Server Access from the Endpoints
  - Map: \{in\_key = server-log-file, in\_value = server-log-id\} ->
    - \{mid\_key = time-period-id, mid\_values = server-log-entries-associated-with-endpoint-id\}
  - GroupBy = time-period-id
  - Reduce: \{mid\_key = time-period-id, mid\_values\} ->
    - \{out\_key = time-period-id, out\_value = List of server access logs from that endpoint, within that time period\}
Step 4a – Clustering Server Accesses

- Cluster the Server Access into
  - Access Type
  - Server/Host URL
  - Frequency of Access

- Filtering out strong clusters
  - Indicative of popular websites, or chatty protocol sessions

- Identify outliers of the clusters
  - Indicative of one-off or reduced set of Server Accesses from the Endpoints
Step 4b – Correlating Server Accesses with Alerts

- Correlate with Un-resolved Alerts
  - Identify strong time correlation between the set of outlier server access logs and Un-resolved Alerts
  - Select Strong correlations as indicators of Alerts to look into
Step 5 – Clustering across correlated Alert-Server Accesses across Endpoint

- Repeat earlier steps for different time-intervals across all the selected Endpoints
- Include correlated Alert-Server Access from various Endpoints into a single data-set
- Clustering to identify commonality across Endpoints
  - Is the Alert-Server Log Correlation seen across multiple Endpoints
  - Identify a Cluster representative Alert-Server Log Correlation for a single Endpoint
- Output the Cluster representative for CIRT Analysis
Step 6 – Involve Human CIRT members (finally)

- Analyze the reported Log-Alert correlations for cluster representative Endpoints
  - Includes looking at the event, the server-access logs and usual CIRT analysis within that time-window

- Feedback results into the Analysis System
  - Negative – Ignore the entire cluster across all the endpoints
  - Positive
    - Select more events from the Cluster for CIRT consumption
    - Select Alerts from that time-window for that particular Endpoints involved with those events

- Rinse and Repeat
Pros

- **Generic Algorithm**
  - No details of specific incident required
  - Can be tweaked
- **Brute Force approach**
  - Can be exhaustive, instead of specific search criteria
- **Data-driven**
- **Fast**
  - Takes advantage of Parallel hardware
- **Can be checked for convergence**

Cons

- Not “Intelligent”
  - May waste CPU cycles
Big Picture for Big Data for Incident Response

Full Packet Capture Data
Net-flow Behavioral Analysis
Automated Sandbox Execution Behavior learning
Asset Configuration
External Information - CVEs
Conclusion

- Big Data seems *made-to-order* for Security Analysis
  - Incident Response is the Killer App
- Big Data-based Machine Learning for Security
  - Moving from Academia to Industry setting
  - Accessible set of OS and Commercial tools
- Big Data for Security – Scope for Innovations
  - Security specific languages, tools
  - OS Frameworks built using existing tools
- Big Data for Security too Big
  - to be ignored by vendors and practitioners