Cybersecurity
Secure the Vision

Threat Hunting through Data Mining and Analytics
Scott Rodgers & Joel Amick

October 2018

Disclosure
This material is for informational purposes only and should not be regarded as a recommendation or an offer to buy or sell any product or service to which this information may relate. Certain products and services may not be available to all entities or persons. Past performance does not guarantee future results.
Who We Are

Joel Amick  
Director  
Cyber Analytics

Scott Rodgers  
Sr. Info Security Analyst  
Cyber Analytics
The vision of Andrew Carnegie

397 offices

$1 trillion¹
assets under management

over 17,500 employees²

Serving 5M individuals

15,000 Institutions serviced by TIAA

As of 09.31.2017
What is Threat Hunting?

Threat Hunting is NOT:

• Using Static Tools/Black Boxes
• Reacting to a Dashboard or Console
• Monitoring Alerts
• A Single State

Threat Hunting IS:

• Proactive
• Hypothesis Based
• Timed
• Measureable
• Collaborative
Threat Hunting Framework Overview

- Hypothesis Formalization
- Source Data
- Outcomes
- Hunting
Requirements

- Access to required data
- Adequate understanding of data and data correlation methods

Hypothesis Criteria

- Specific
- Provable/disprovable
- Does not necessarily need to be if/then scenario

Ethical Concerns

- Partner with Human Resources and Legal
Hypothesis Creation

Example of Poor Hypothesis

“We have web account registration fraud on our website.”

Example of Good Hypothesis

“There is a statistically significant amount of web account registration fraud that can be identified by comparing IP geolocation and time zone.”
Hunt Team

Data Scientist/Analyst
- Statistical Background
- Data Mining
- Data Access

Cybersecurity Analyst/Investigator
- Subject Matter Expert
- Develop Hypotheses
- Validating Results
External Threat

Freddy Fraud gathers stolen identity data to try to register fraudulent accounts on various websites.
Freddy has recently discovered your organization and decides to try a cache of stolen identity on your organization’s website.

Insider Threat

Ivy Insider works as a developer at your organization. Recently, she was granted permissions that enables her to log on to any workstation and view the hard drive contents.
Ivy decides to seize this opportunity to log on to a number of executive machines to look for trade secret information to sell.
Hunt Walkthrough – External Threat

**Target**

Web Account Registration Fraud

**Hypothesis**

Mismatched attributes observed during web account registration have a statistically significant rate of fraud.

**Source Data**

- Web Account Registration Data
- Web Traffic Data
- Participant Data

**Hunt Steps**

- Identify Correlating Attributes
- Join and Enrich Datasets
- Create Additional Fields / Features

**Outcomes**

- Actionable Intelligence
  - Rule Creation
  - Learning
Hunt Process

Gather Data
- Participant Data
- Web Traffic Data
- Account Creation Data

Cleaning Data
- Clean Columns
- Dedup Data

Merging Data

Recursive
- Feature Generation
- Feature Elimination (RFE)
- Fuzzy String Matching
Python Pseudo Code

```python
## Read Participant CSV
part_data = pd.read_csv(file_path)

## Get Web Account Registration Data
conn = pyodbc.connect(connection_string)
sql = "Select * from db.account_creation"
acc_data = pd.read_sql(sql, conn)

## Drop Duplicates
acc_data = acc_data.drop_duplicates(subset="user_id", keep="first")

## Merge
merged_data = pd.merge(part_data, acc_data, left_index=True, right_index=True, sort=False)

## Calculate Fuzzy String Match Score
matched_data = check_matches(merged_data, merged_col_names)

## Save to CSV
matched_data.to_csv("Merged Data", index_label="user_id")
```

Ref: https://pandas.pydata.org/
## Read Participant CSV
part_data = pd.read_csv(file_path)

## Get Web Account Registration Data
conn = pyodbc.connect(connection_string)
sql = "Select * from db.account_creation"
acc_data = pd.read_sql(sql, conn)

## Drop Duplicates
acc_data = acc_data.drop_duplicates(subset="user_id", keep="first")

## Merge
merged_data = pd.merge(part_data, acc_data, left_index=True, right_index=True, sort=False)

## Calculate Fuzzy String Match Score
matched_data = check_matches(merged_data, merged_col_names)

## Save to CSV
matched_data.to_csv("Merged Data", index_label="user_id")
## Read Participant CSV
part_data = pd.read_csv(file_path)

## Get Web Account Registration Data
conn = pyodbc.connect(connection_string)
sql = "Select * from db.account_creation"
acc_data = pd.read_sql(sql, conn)

## Drop Duplicates
acc_data = acc_data.drop_duplicates(subset="user_id", keep="first")

## Merge
merged_data = pd.merge(part_data, acc_data, left_index=True, right_index=True, sort=False)

## Calculate Fuzzy String Match Score
matched_data = check_matches(merged_data, merged_col_names)

## Save to CSV
matched_data.to_csv("Merged Data", index_label="user_id")

## Drop Duplicates
acc_data = acc_data.drop_duplicates(subset="user_id", keep="first")
## Read Participant CSV
part_data = pd.read_csv(file_path)

## Get Web Account Registration Data
conn = pyodbc.connect(connection_string)
sql = "Select * from db.account_creation"
acc_data = pd.read_sql(sql, conn)

## Drop Duplicates
acc_data = acc_data.drop_duplicates(subset="user_id", keep="first")

## Merge
merged_data = pd.merge(part_data, acc_data, left_index=True, right_index=True, sort=False)

## Calculate Fuzzy String Match Score
matched_data = check_matches(merged_data, merged_col_names)

## Save to CSV
matched_data.to_csv("Merged Data", index_label="user_id")
## Read Participant CSV
part_data = pd.read_csv(file_path)

## Get Web Account Registration Data
conn = pyodbc.connect(connection_string)
sql = "Select * from db.account_creation"
acc_data = pd.read_sql(sql, conn)

## Drop Duplicates
acc_data = acc_data.drop_duplicates(subset="user_id", keep="first")

## Merge
merged_data = pd.merge(part_data, acc_data, left_index=True, right_index=True, sort=False)

## Calculate Fuzzy String Match Score
matched_data = check_matches(merged_data, merged_col_names)

## Save to CSV
matched_data.to_csv("Merged Data", index_label="user_id")

## Calculate Fuzzy String Match Score
def check_matches(data, col_names):
    for combo in combinations(cols, 2):
        compare_loop(data, combo[0], combo[1])

def compare_loop(data, col1, col2):
    for index, row in Data.iterrows():
        fuzz.token_sort_ratio(row[col1], row[col2])

Ref: https://github.com/seatgeek/fuzzywuzzy
## Read Participant CSV
part_data = pd.read_csv(file_path)

## Get Web Account Registration Data
conn = pyodbc.connect(connection_string)
sql = "Select * from db.account_creation"
acc_data = pd.read_sql(sql, conn)

## Drop Duplicates
acc_data = acc_data.drop_duplicates(subset="user_id", keep="first")

## Merge
merged_data = pd.merge(part_data, acc_data, left_index=True, right_index=True, sort=False)

## Calculate Fuzzy String Match Score
matched_data = check_matches(merged_data, merged_col_names)

## Save to CSV
matched_data.to_csv("Merged Data", index_label="user_id")
## What Are We Looking For?

### Customer Provided Email Domain
- @zzz.zz
- @TIAABank.org
- @TIAA.org

### Organization Provided Email Domain
- @TIAA.org
- @TIAA.org
- @TIAA.org

### Fuzzy String Match Percentage
- Low Attribute Similarity %: 14%
- High Attribute Similarity %: 80%
What Are We Looking For?

Ideal Distribution for an Alert

- **Fraud**
- **Non-Fraud**

Alert Threshold

Low Attribute Similarity %

High Attribute Similarity %
What Did We Find?

**Fraud Identified**

- **Alert Threshold**
- **Fraud**
- **Non-Fraud**

**Ideal Distribution for an Alert**
- Low Attribute Similarity %
- High Attribute Similarity %

**Hypothesis Formalization**
- **Source Data**
- **Hunting**
- **Outcomes**
What Would a Bad Example Look Like?

Alert Threshold?

Fraud
Non-Fraud

Low Attribute Similarity %
High Attribute Similarity %

Ideal Distribution for an Alert

Alert Threshold
Fraud
Non-Fraud

TIAA PUBLIC
Hypothesis – Proven

- Mismatched attributes (Email Address Domain – Customer Organization Domain) observed during web account registration have a statistically significant rate of fraud.

Actionable Intelligence

- Additional fraudulent web account registration events identified by mismatched attributes.

Policy Issues Identified

- Why are users allowed to register a new web account with an email domain that does not match their organization?

Learning

- Hunters gained hands-on experience using and correlating web account registration data.
Users with elevated permissions are accessing executive workstations without business justification.

**Source Data**
- Windows Event Logs
- Employee Titles
- Workstation Data
- Support Ticketing Data

**Hunt Steps**
- Identify Correlating Attributes
- Join and Enrich Datasets
- Investigate Results

**Outcomes**
- Report Actionable Intelligence
- Identified Policy Issue
- Developed Mitigating Alerting Strategy
1. Select the columns:
2. `Select w.WorkstationName, u.UserID, u.Name, u.JobTitle` from `db.Users u`  
3. Join Workstation Data: `join db.workstation w` on `u.UserID=d. UserID`  
4. Filter based on Job Title: `where u.JobTitle Like '%exec%' OR u.JobTitle Like '%vp%'`

<table>
<thead>
<tr>
<th>Workstation Name</th>
<th>User ID</th>
<th>Name</th>
<th>Job Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>exec-pc-001</td>
<td>exec001</td>
<td>Eric</td>
<td>Executive</td>
</tr>
<tr>
<td>exec-pc-002</td>
<td>exec002</td>
<td>Eddie</td>
<td>Executive</td>
</tr>
<tr>
<td>exec-pc-003</td>
<td>exec003</td>
<td>Erin</td>
<td>Executive</td>
</tr>
<tr>
<td>vp-pc-005</td>
<td>vp005</td>
<td>Val</td>
<td>VP</td>
</tr>
</tbody>
</table>
An account was successfully logged on.

Subject:
- Security ID: SYSTEM
- Account Name: EXEC-PC-001$
- Account Domain: WORKGROUP
- Logon ID: 0x3E7

Logon Information:
- Logon Type: 3
  - Restricted Admin Mode: -
  - Virtual Account: No
  - Elevated Token: No

Impersonation Level: Impersonation

New Logon:
- Security ID: AzureAD\exec001$
  - Account Name: exec001@org.com
  - Account Domain: AzureAD
  - Logon ID: 0xFD5113F
  - Linked Logon ID: 0xFD5112A
  - Network Account Name: -
  - Network Account Domain: -
  - Logon GUID: {00000000-0000-0000-0000-000000000000}

Process Information:
- Process ID: 0x30c
- Process Name: C:\Windows\System32\lsass.exe

Network Information:
- Workstation Name: EXEC-PC-001
  - Source Network Address: -
  - Source Port: -

Detailed Authentication Information:
- Logon Process: Negotiate
- Authentication Package: Negotiate
- Transited Services: -
- Package Name (NTLM only): -
- Key Length: 0

Fields used in hunt

Ref: https://www.ultimatewindowssecurity.com
## Look for successful windows login events
`eventtype=microsoft-windows-events AND eventcode=4624`

## Filter logins to executive workstations
`AND (workstation_name=exec-pc-* OR workstation_name=vp-pc-005)`

## Aggregate login events by workstation
`| stats count(EventCode) as “Login Count”, dc(Security_ID) as “User Count”, values(Security_ID) as “Users”, values(Logon_Type) as “Logon Type” by workstation_name`

## Only show workstations with more than 1 distinct user
`| where “User Count” > 1`
### Elastic Search Results

<table>
<thead>
<tr>
<th>Computer Name</th>
<th>User Count</th>
<th>Login Count</th>
<th>Users</th>
<th>Title</th>
<th>Logon Type</th>
<th>Logon Type Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>exec-pc-001</td>
<td>2</td>
<td>1</td>
<td>admin_ivy_insider</td>
<td>Developer</td>
<td>3</td>
<td>Network</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>exec001</td>
<td>Executive</td>
<td>2</td>
<td>Interactive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>admin_ivy_insider</td>
<td>Developer</td>
<td>3</td>
<td>Network</td>
</tr>
<tr>
<td></td>
<td></td>
<td>38</td>
<td>Exec002</td>
<td>Executive</td>
<td>2</td>
<td>Interactive</td>
</tr>
<tr>
<td>exec-pc-002</td>
<td>2</td>
<td>1</td>
<td>admin_service_desk_ron</td>
<td>Sr. Desktop Support</td>
<td>10</td>
<td>Remote Interactive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>service_desk_elliot</td>
<td>Desktop Support</td>
<td>10</td>
<td>Remote Interactive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>exec003</td>
<td>Executive</td>
<td>2</td>
<td>Interactive</td>
</tr>
<tr>
<td>exec-pc-003</td>
<td>3</td>
<td>1</td>
<td>admin_service_desk_ron</td>
<td>Sr. Desktop Support</td>
<td>10</td>
<td>Remote Interactive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>service_desk_elliot</td>
<td>Desktop Support</td>
<td>10</td>
<td>Remote Interactive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>exec003</td>
<td>Executive</td>
<td>2</td>
<td>Interactive</td>
</tr>
<tr>
<td>vp-pc-005</td>
<td>2</td>
<td>2</td>
<td>service_desk_elliot</td>
<td>Desktop Support</td>
<td>10</td>
<td>Remote Interactive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23</td>
<td>vp005</td>
<td>Vice President</td>
<td>2</td>
<td>Interactive</td>
</tr>
</tbody>
</table>
Hypothesis – Proven

• Users with elevated permissions are accessing executive workstations without business justification.

Actionable Intelligence

• User Ivy Insider abusing privileges and logging into executive systems.

Policy Issues Identified

• Is least privilege being applied to user permissions?

Learning

• Correlated disparate data & built understanding of data
Hunt Ideas

External Threat
- Web Account Creation
  - Web Channel
  - Phone Channel
- Account Takeover
- Cross-Channel Fraud
- One Time Pin (OTP) Abuse

Insider Threat
- Unapproved or Portable Applications
- Personal VPN Clients to avoid Data Loss Prevention Tools
- File Sharing
- Remote Desktop / Access Tools
Key Takeaways

- Delivering Actionable Intelligence
- Validating Investigators Hypotheses
- Iterating with Investigators to Improve
- Driving Policy Changes
- Proactive vs Reactive
Q&A