.conf2015

From Zero to Pretty Robust Fraud Detection Tool

Tomasz Dziedzic

CTO, Linux Polska

splunk>

Disclaimer

During the course of this presentation, we may make forward looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC. The forward-looking statements made in the this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward looking statements we may make.

In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not, be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

Personal Introduction

- Tomasz Dziedzic, CTO @ Linux Polska
- Integration and Consulting Services for Splunk and Data Science
- How we got here and acknowledgements
- Splunk Team @ Linux Polska
- Dariusz Kwaśny – Project Lead for Fraud Detection Tool

Agenda

I will walk you through a story of fraud detection effort made by an investment bank from Poland.

They took pretty straightforward steps to address fraudulent activities and save $M in the process.

Key take-aways:

- *Quick and dirty searches – Go for fast results under pressure*
- *(ab)Use KVStore – optimize performance as you gain speed*
- *Combining data sources is always a good idea!*
- *Tools and application change with every new bit of knowledge (or die!)*

The Story

- One of the bank's customers filed a complaint regarding failed transfer that did not reach the intended recipient.
- Investigation started with disparate teams working in parallel.
- Our team applied application log analysis using Splunk.
- After 2 days of effort the initial solution was formulated and first few $M saved in the process.

Quick and Dirty Solution to the Problem

```
index=fds   source=app_log  ActionRemit.java INFO SES_*_*_* | rex field=_raw ",\d+\s\[\s*(?<sessid>[^\]]+)\]" | rex field=sessid "SES_(?<login>[^_]
+)_" | rex field=_raw "Customer account:\s(?<customer_account>\d+)" | rex field=_raw "Recepient\saccount:\s(?<rcpt_account>\d+)" | rex
field=_raw "Amt:\s(?<amount>[\d\.]+)" | rex field=_raw "RemitId:\s(?<id_remit>\d+)" | rex field=_raw "Remit\sTitle:\s(?<remit_title>.+)" | rex
field=_raw "Recipient\sName:\s(?<rcpt_name>.+)" | rex field=_raw "Date:\s(?<remit_date>.+)" | transaction sessid startswith="ActionRemit.java:
610" | join sessid [search index=fds | search  NOT ("/images" OR "/styles" OR DEBUG) | search ActionRemit OR ActionStart OR AuthAgentService |
rex "(for host|from host)(?<srcipaddr>[\d\.]+)(\s|$)" | rex ",\d+\s\[\s*(?<sessid>[^\]]+)\]" | rex field=sessid "SES_(?<sessid2>[^_]+_\d+)_" | rex
field=sessid "SES_(?<login_app>[^_]+)_\d+_" | rex "User\s(?<login_app_logged>[^\s]+)\slogged" | eval login_app_logged=upper(login_app_logged)
| rex "AuthAgentService\sreports:\sUser\s\"(?<aas_username>[^\"]+)\"" | eval aas_username=upper(aas_username) | eval user=upper(user) | eval
login=coalesce(login_app, login_app_logged, aas_username, user) | rex "Customer\saccount:\s(?<customer_account>\d+)" | rex "Recepient
\saccount:\s(?<rcpt_account>\d+)" | rex "Amt:\s(?<amount>[\d\.]+)" | rex "RemitId:\s(?<id_remit>\d+)" | rex "Remit\sTitle:\s(?<remit_title>.+)" |
rex "Recipient\sName:\s(?<rcpt_name>.+)" | transaction login sessid2   startswith="*AuthAgentService reports:*was loged-in" connected=f | search
actionRemitSave | stats count by _time, sessid, srcipaddr | table sessid, srcipaddr] | stats list(rcpt_account) as "Recipient Account", list(sessid) as
"Session ID", list(srcipaddr) as "IP Address", list(id_remit) as "Remit ID", list(login) as "Login", values(remit_date) as "Remit Date", list(_time) as tm,
dc(rcpt_account) as "numRcptAccounts"  by customer_account, amount, remit_title, rcpt_name | eval "Event Time"=strftime(tm, "%d.%m.%Y %H:
%M:%S") | eval severity_amount=case(amount<20000,1, amount<500000,3, amount>=500000,6) | eval
severity_rcpt_account=case(numRcptAccounts==2,6, (numRcptAccounts==3 OR numRcptAccounts==4),3, numRcptAccounts>4, 1) | eval
Severity=severity_amount+severity_rcpt_account | rename  amount AS "Remit Amount", remit_title as "Remit Title", rcpt_name as "Recipient
name" | where numRcptAccounts>1 | lookup userslookup customer_account OUTPUT NAME1 NAME2 NAME3 NAME4 CUSTOMER_ID | eval
"Customer Name"=if(isnotnull(NAME2), NAME1." ".NAME2, NAME1) | eval "Customer Address"=if(isnotnull(NAME4), NAME3.", ".NAME4, NAME3) |
rename CUSTOMER_ID as "Customer ID" | table  "Customer Name" "Customer Address"  "Remit Amount" "Remit Title" "Remit ID" "Remit Date"
"Recipient name" "Recipient Account" "Event time" "Session ID" "Login" "IP Address", "Severity" | sort - "Event time
```

And the neat List of Suspicious Activity

Suspicious Activity

Edit | More Info

1m ago

Customer Name	Customer Address	Remit Amount	Remit Title	Remit ID	Remit Date	Recipient name	Recipient Account	Event time	Session ID	Login	IP Address	Severity
Charlie's Chocolate Factory	Drury Lane Theatre | Catherine Street London WC2B 5JF England	1000000	For cocoa delivery	1234	2015-03-01 12:15	Extremly Big Cocoa Factory	2456760030303 2750394860303456	2015-03-01 13:25 2015-03-01 12:15	3234564 123456	CHARLIE123 CHARLIE123	123.9.8.5 125.4.2.5	12
Charlie's Chocolate Factory	Drury Lane Theatre | Catherine Street London WC2B 5JF England	500000	For another cocoa delivery	3354	2015-03-01 14:10	Extremly Big Cocoa Factory	2750394860303456 2780354840303457	2015-03-01 16:05 2015-03-01 14:10	77888 77777	CHARLIE555 CHARLIE555	125.4.2.5 125.4.2.5	6

$

Account numbers

Source IPs

Primitive scoring

First Lessons

- Start solving the problem with limited understanding and mature on the way over.
- Focus on the most wanted issues and make shortcuts if needed.
- Leverage very tight improvement (feedback) loop (plan, do, check, act).
- Engage in communication despite challenging environment and pressure.
- Excellent ROI despite being: rough, inefficient, absorbing.

Don't hesitate. Start now!

First Weeks with New capability

Some frauds were prevented resulting in several $M saved just within first weeks.

There were numerous issues though:

- High false positive ratio.
- High operator overhead for fraud verification / investigation.
- Low fraud pattern coverage.
- Low operator flexibility.
- After initial WOW factor wore off it suddenly became slow!

.conf2015

There are Better Ways to do it!

splunk>

What Do We Need?

- Generic tool with a more accurate user activity profiling
- Visibility and usability
- Filter out the safe cases
- Highlight the obvious threats
- Configuration flexibility

What Do We Have?

- The idea for a way more efficient rules
- Application logs
- Web access logs
- Account whitelist
- Account blacklist

Let's Get to Business

better visibility + adequate rules = success recipe 2.0

If we could profile the whole user session for actions

AND

If we could track the source IP

AND

If we haven't seen this IP yet in other sessions/activities older than X

THEN

We should look into this activity

Application Log that Contains User Actions Trace (1/2)

application log (sourcetype=app_log)

```
2015-03-02 13:22:55,625 [        SES_92264_2](    ActionStart.java:  98) DEBUG : User AAAAAA logged in, access to services: '000110000'
2015-03-02 13:22:55,693 [SES_BBBBBB_88864_14](    EJBManager.java:311) DEBUG : getCurrencyList()
2015-03-02 13:22:55,694 [SES_BBBBBB_88864_14](  ActionXYZ.java:473) INFO  : setCurrencyList()
2015-03-02 13:22:55,695 [        SES_89874_2](    ActionStart.java: 120) DEBUG : Very detailed debug message
2015-03-02 13:22:55,762 [        SES_89874_2](    ActionStart.java: 116) INFO  : Informational info message
2015-03-02 13:22:55,821 [SES_CCCCCC_91026_19](  ActionAccount.java: 437) DEBUG : execute() something
……
```

Authenticated User ID

Session ID right before successfull authentication

Application Log that Contains User Actions Trace (2/2)

application log (sourcetype=app_log)

```
.......
2015-03-02 13:23:56,220 [SES_AAAAAA_92264_3](  ActionRemit.java: 610) INFO  : Execute AcionRemit()
2015-03-02 13:23:56,220 [SES_AAAAAA_92264_3](  ActionRemit.java: 610) INFO  : ElxSorb: 0
2015-03-02 13:23:56,221 [SES_AAAAAA_92264_3](  ActionRemit.java: 609) INFO  : Status: AI
2015-03-02 13:23:56,222 [SES_AAAAAA_92264_3](  ActionRemit.java: 608) INFO  : Remit Type: 1
2015-03-02 13:23:56,223 [SES_AAAAAA_92264_3](  ActionRemit.java: 607) INFO  : Recipient account: 11111122222233333444445555
2015-03-02 13:23:56,224 [SES_AAAAAA_92264_3](  ActionRemit.java: 606) INFO  : Recipient bank: 22223333
2015-03-02 13:23:56,225 [SES_AAAAAA_92264_3](  ActionRemit.java: 605) INFO  : Amt: 100756.00
2015-03-02 13:23:56,233 [SES_AAAAAA_92264_3](  ActionRemit.java: 603) INFO  : Recepient Name: Ben Johnson, Baker Street 666
2015-03-02 13:23:56,244 [SES_AAAAAA_92264_3](  ActionRemit.java: 603) INFO  : Recepient Title: Salary for two months
2015-03-02 13:23:56,255 [SES_AAAAAA_92264_3](  ActionRemit.java: 601) INFO  : Date: 2015-03-02
2015-03-02 13:23:56,277 [SES_AAAAAA_92264_3](  ActionRemit.java: 599) INFO  : AccountId: 1893211
2015-03-02 13:23:56,291 [SES_AAAAAA_92264_3](  ActionRemit.java: 598) INFO  : RemitId: 79304
```

Session ID after successfull authentication

Web Frontend Log: Application Access

web server log (sourcetype=error)

[02/Mar/2015:13:22:55] info (29928): for host 190.50.60.1 trying to POST /, AuthAgentService reports: User "AAAAAA" was logged-in

[02/Mar/2015:13:22:55] info (29928): for host 190.50.60.1 trying to POST /, IsCredentialsValid reports: Ace stub: securid/check ret code: 0, user: ”AAAAAA"

[02/Mar/2015:13:22:56] info (6385): Auth statistics reports: On-line users: 34

[02/Mar/2015:13:22:56] info (6385): for host 93.234.234.2 trying to POST /, AuthAgentService reports: User ”DDDDDD" was logged-in

[02/Mar/2015:13:22:56] info (6385): for host 93.234.234.2 trying to POST /, IsCredentialsValid reports: Ace stub: securid/check ret code: 0, user: ”DDDDDD"

[02/Mar/2015:13:22:56] info (7386): Auth statistics reports: On-line users: 36

…….

[02/Mar/2015:13:30:58] info (7386): for host 190.50.60.1 trying to GET /.auth, AuthAgentService reports: User "AAAAAA" was logged-out

[02/Mar/2015:13:45:58] info (19748): InternalCheck reports: Session expired for user “EEEEEE” connected from host 10.2.3.4

Source IP for User ID and session

Search to Profile User Sessions

And highlight those from previously “unseen” source IPs

```
index=fds (sourcetype=error OR sourcetype=app_log) NOT ("/images" OR "/styles")
| search AuthagentService OR InternalCheck OR ActionStart $filter_action$
| eval login=coalesce(login, login_expired,  login_app_action_start, login_app)
| transaction maxevents=3000 connected=t $filter_keepevicted$ login sessid
                    startswith="logged-in" endswith="expired OR logged-out”
| search login=$login$ sessid=$sessid$
| rex "(for host|from host)(?<ip>[\d\.]+)(\s|$)„
| join type=outer login, ip [ inputlookup fds_ip_lookup ]
| eval new_ip=if(isnull(count),"yes","no")
| search new_ip=$filter_new_ip$
```

filter initial data

normalize login fields

build sessions

filter interesting sessions or logins

extract source IPs

add successfull login count for login + source IP

enable filtering for source IP first-timers

fds_ip_lookup configuration

$SPLUNK_HOME/etc/$APP_NAME/local/collect.conf

```
[fds_ip]
enforceTypes = true
field.count = number        # how many successful logins?
field.ip = string           # src ip address
field.login = string        # login
```

Lookup and collection definition

Config files

$SPLUNK_HOME/etc/$APP_NAME/local/transforms.conf

```
[fds_ip_lookup]
collection = fds_ip
external_type = kvstore
fields_list = ip, login, count
```

Populate and Update fds_ip_lookup

```
index=fds latest=-30d sourcetype=error
| stats count by login, srcip
| outputlookup fds_ip_lookup
```

There is highly unlikely (this case specific rule) that IPs seen more than 30 days ago are used by attackers

Resulting output is used to refresh KVStore collection

Add Black and White List Information

```
| lookup accounts_blacklist recipient_account OUTPUT recipient_account as recipient_account_bl
| eval bl_status=if(isnotnull(recipient_account_bl),"onBlackList","notOnBlackList")
```

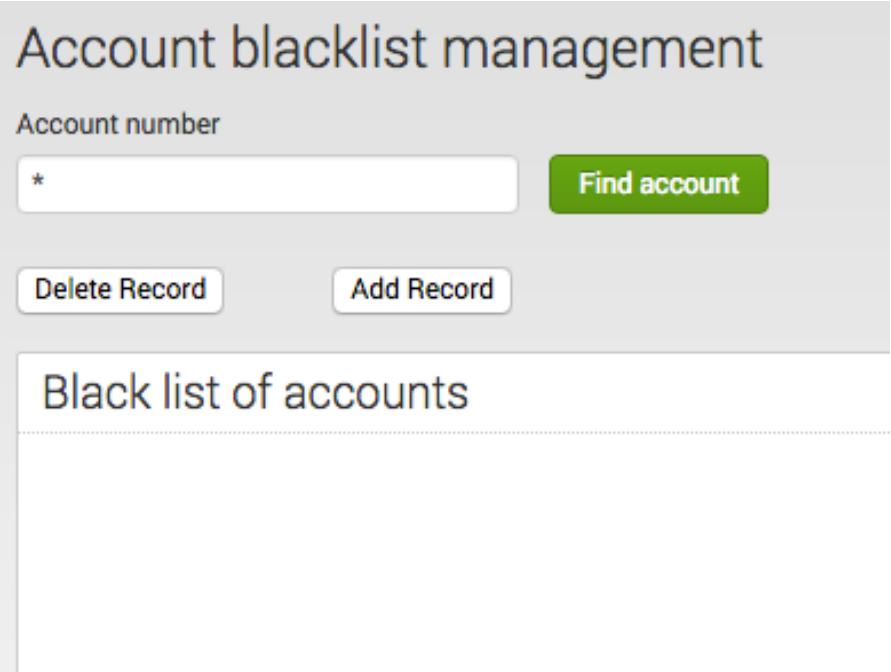
Combine your list information with transaction data

*_status is used to discern and make proper actions later

```
| lookup accounts_whitelist recipient_account OUTPUT recipient_account as recipient_account_wl
| eval wl_account_status=if(isnotnull(recipient_account_wl),"onWhiteList","notOnWhiteList”)
```

Simple Account List Management UI

- Just enough to enable management
- Built using standard Splunk UI elements
- Plans to add Javascript field validation to avoid faulty account numbers in the future



Lower RDBMS Utilization

Customer details lookups are accelerated using KVStore

Runs once a day to update
the KVStore collection

```
| dbquery customers_accounts_db "SELECT ID_ACCOUNT, NAME1, NAME2 FROM prod.customers”
| outputlookup clients_kvs_lookup

| lookup clients_kvs_lookup ID_ACCOUNT OUTPUT NAME1 NAME2
| eval customer_name=if(isnotnull(NAME2), NAME1." ".NAME2, NAME1)
```

KVstore collection is used
to supplement the transaction details

Success 3.0 and Beyond

- Algorithms based scoring for faster resolution
- Automatic transfer suspending for high score events
- Integration with external blacklist sources
- UI enhancements for Operator / Investigator productivity
- Reporting for wider business audience

.conf2015

Practice at Home

splunk>

Approach to Maximize ROI

Workshop style engagement

Look for one of three key factors:

- Most troublesome / infamous
- Most money / daily business involved
- Most exposed / essential

If a name rings more than once you have a good candidate!

Go for vertical monitoring / analysis. Pick one system or process and model all the way up.

Requires breaking the silos boundaries!

You have been warned!

Linux Polska Splunk Capabilities

Now playing:

End to End monitoring for Space Industry

App for JBoss

Add-on for JBoss

Add-on for Postgres

App for Postgres

OpenStack API Technology Add-on

OpenStack Horizon Technology Add-on

OpenStack Nova Technology Add-on

OpenStack Keystone Technology Add-on

Retail Intelligence

Main purpose of Retail Intelligence App is to bring meaningful, actionable data to marketers responsible for traditional in-store and on-line sell. Basic unit of analysis is purchasing

Content: App | **Categories:** Business Analytics | **Author:** Linux Polska | **Downloads:** 4 | **Released:** Sep 01, 2015 | **Updated:** Sep 01, 2015

Infrastructure & Middleware & Cloud

Business oriented

Available on ***splunkbase***

How to Make the Most of it

Get the data

Focus on Your challenge

Make a positive change

Again!

Questions?

THANK YOU