Copyright © 2015 Splunk Inc.

Detecting Bank Accounts Takeover Cyber Attacks with Splunk >

Gleb Esman

Consultant, Financial Services Firm

Since 08/15 Sr. Product Manager, Splunk

splunk>

Copyright © 2015 Splunk Inc.

.conf2015

Detecting Bank Accounts Takeover Cyber Attacks with Splunk >

Session materials, raw code snippets, articles and updates are available at my blog at:

http://www.mensk.com

splunk>

.conf2015

”Better to learn about your fraud events from Splunk than from CNN”

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.

Bonus Disclaimer

I cannot disclose any specifics about operations of security divisions of any clients.

Session Purpose

To teach how Splunk can be used as a custom security tool to detect account takeover cyber attacks.

Benefits

- Near real time detection of ATO attempts
- Proven and moved into production deployment
- Suitable for any type of business with web presence
- Quick (~2-3 weeks) from zero to deployment
- Works with any type of data formats

Introduction

- In a number of cases the Splunk-based solution described here was able to detect ATO attacks better than any other commercially deployed anti-fraud security system.

- This solution described can be applied to many industry verticals and enterprises**: financials, e-commerce, insurance, healthcare, education and others** to provide an extra layer of protection to clients and customer accounts.

Abstract

- What is Fraud? What is Account Takeover attack?
- Define task to detect account takeovers with Splunk
- Implement summary index of logins data
- Build final scheduled search to detect ATOs
- Step by step explanations
- Minimizing false positives
- **Bonus:** Detect more sophisticated attacks with Splunk!

Personal Introduction

- 1990's: Israeli anti-virus research and development company
- 2000's: IBM T. J. Watson Research Center / anti-virus / development of heuristic anti-malware virtual machines to detect known and unknown malware
- Till July, 2015 - Consultant, Financial Services Firm, Montreal
- Since August, 2015 – Sr. Product Manager at Splunk / Anti-Fraud Products, San Francisco
- While at Financial Services Firm:
  - Leading an effort to utilize Splunk as anti-fraud platform for online banking.
  - Single screen fraud investigation dashboard with detailed drilldowns.
  - Custom fraud analytics, threat intelligence feeds correlation and ATO detection.
  - Managing firm-wide IBM Tealeaf Analytics system deployment and working closely with strategic security and e-Fraud teams to build solutions helping to detect and investigate fraud.
  - Opportunity to build highly tuned security solutions on top of a mountain of financial and retail banking data.

What is **Fraud**?

Deception

Intentional Deception

Intentional Deception of a person or entity

Intentional Deception of a person or entity by another

Intentional Deception of a person or entity by another made for gain

Intentional Deception of a person or entity by another made for monetary or personal gain

Intentional Deception of a person or entity by another made for monetary or personal gain or to cause a loss to another party

What is **Account Takeover**?

Account takeover fraud occurs when **someone** other than the authorized account holder **gains access to an existing account**.

Consequences of Account Takeover Cyber Attacks:

- Identity theft, personal and confidential information loss
- Damages to client confidence levels
- Significant business losses due to litigation
- Hurtful to brand integrity, business and industry reputation
- Losses due to business disruption to mitigate consequences
- Significant monetary damages

What's Happening

- Valid clients credentials fallen into hands of bad guys through phishing attacks, malware, spyware. Resold on a black markets
- Bad guys are trying to take over clients accounts and steal money (via wire transfers, fraudulent bill payments), run fraudulent securities trading (pump and dump schemes)
- Best time to **neutralize** attack is ***at the first line of defense*** – detecting attacker during the initial login attempts before damage is done
- Why Splunk? Access to all data already

How to Detect ATO ASAP?

What attackers do?

When batch of client credentials leaks into black markets attackers will start testing credentials and quite often multiple user accounts are accessed from the same IP address or subnet

What do we need to detect and stop attacks?

- Data sources – as close to real time as possible
- Regular IIS/Apache/Web logs might not be enough – need ***username*** field present + HTTP headers, cookies data
- Ideal: Splunk Stream or similar real time data sources (Extrahop)

Setting up Automated Alerting on ATO

Definition of task, Iteration 1

- **Alert if**:
  Multiple clients accounts are being "touched" by an unknown IP address (never used by the client) *and* by an unknown browser (USER_AGENT never used by the client).
- **Potential problems**:
  - Attacker may try to change IP address frequently. Easiest to change IP within the same subnet: 123.45.6.0/24
  - Attacker might happen to use the same browser that legitimate client does
  - False positives: financial services, aggregators, corporate accounts might "behave" in the same way as attacker does
  - Need to have clearly defined time ranges and trigger limits, at least initially

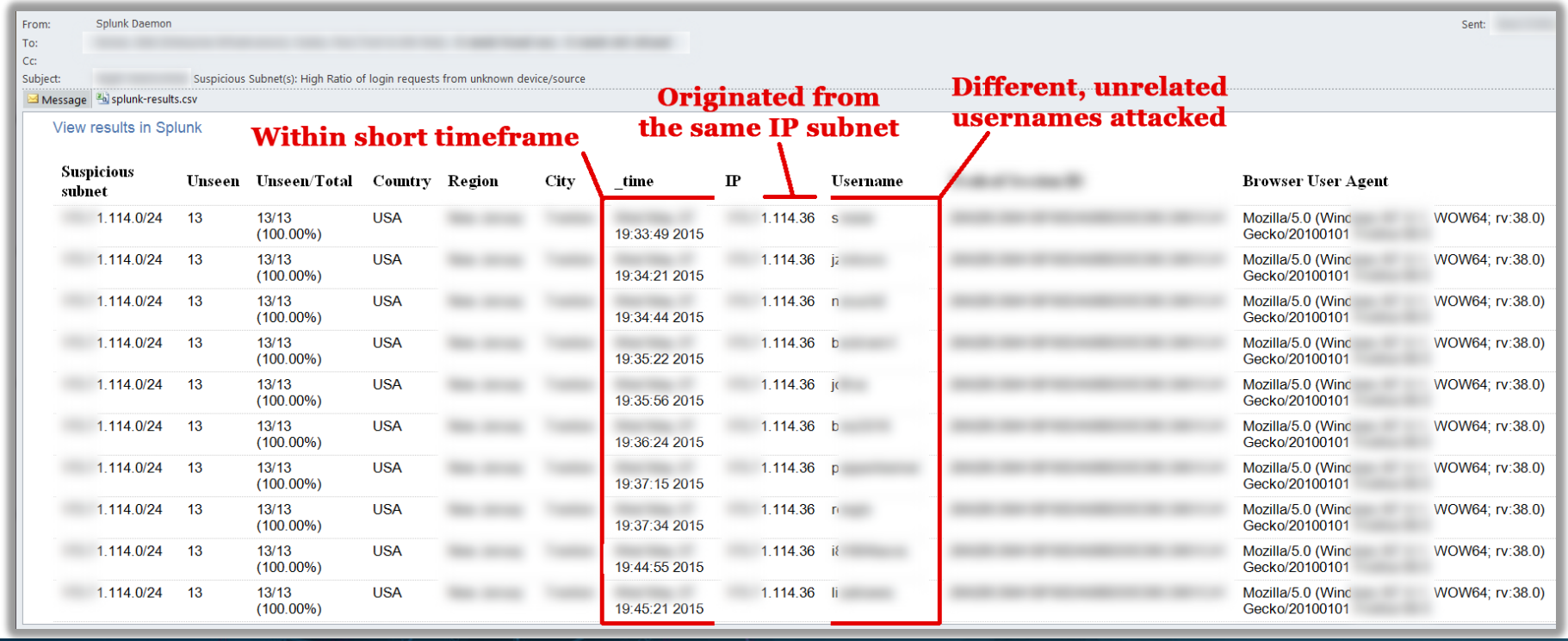
Setting up Automated Alerting on ATO

Definition of task, Iteration 2

- **Alert if**:
  At least 5 client accounts are being "touched" by the same IP subnet (N.N.N.0/24) within 1 hour -
  AND at least 75% of total accounts touched have never been accessed from this subnet *and* never by this browser (USER_AGENT) within the last 45 days.
- *"Touched" means attempted to log into, regardless of success.*
- ~~**Potential problems**~~:
  - ~~Attacker may try to change IP address frequently~~ **using subnet instead of fixed IP**
  - ~~Attacker might happen to use the same browser~~ **allowing for 25% browser match**
  - ~~False positives: financial services, aggregators~~ … **will utilize specific whitelists**
  - ~~Need to have clearly defined time ranges and trigger limits~~ **done**

Setting up Automated Alerting on ATO

How ATO email alert looks like:



Suspicious subnet	Unseen	Unseen/Total	Country	Region	City	_time	IP	Username	Browser User Agent
1.114.0/24	13	13/13 (100.00%)	USA			19:33:49 2015	1.114.36	s	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:34:21 2015	1.114.36	j	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:34:44 2015	1.114.36	n	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:35:22 2015	1.114.36	b	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:35:56 2015	1.114.36	j	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:36:24 2015	1.114.36	b	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:37:15 2015	1.114.36	p	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:37:34 2015	1.114.36	r	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:44:55 2015	1.114.36	i	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:45:21 2015	1.114.36	li	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101

Setting up Automated Alerting on ATO

Implementation of task

Splunk query needs to:

1. Scan most recent **1 hour** of access log data and find the list of subnets that tried to access multiple (**>=5**) accounts within that hour.
2. For each of these accounts – take username, IP, USER_AGENT and scan previous **45 days** of traffic history to find if any of these usernames has **never been touched** by this IP/USER_AGENT combo.
3. Alert if number of found accounts is above **75%** threshold per subnet.

Problem: #2 will take prohibitively long time.

Solution: need to build summary index of client logins events

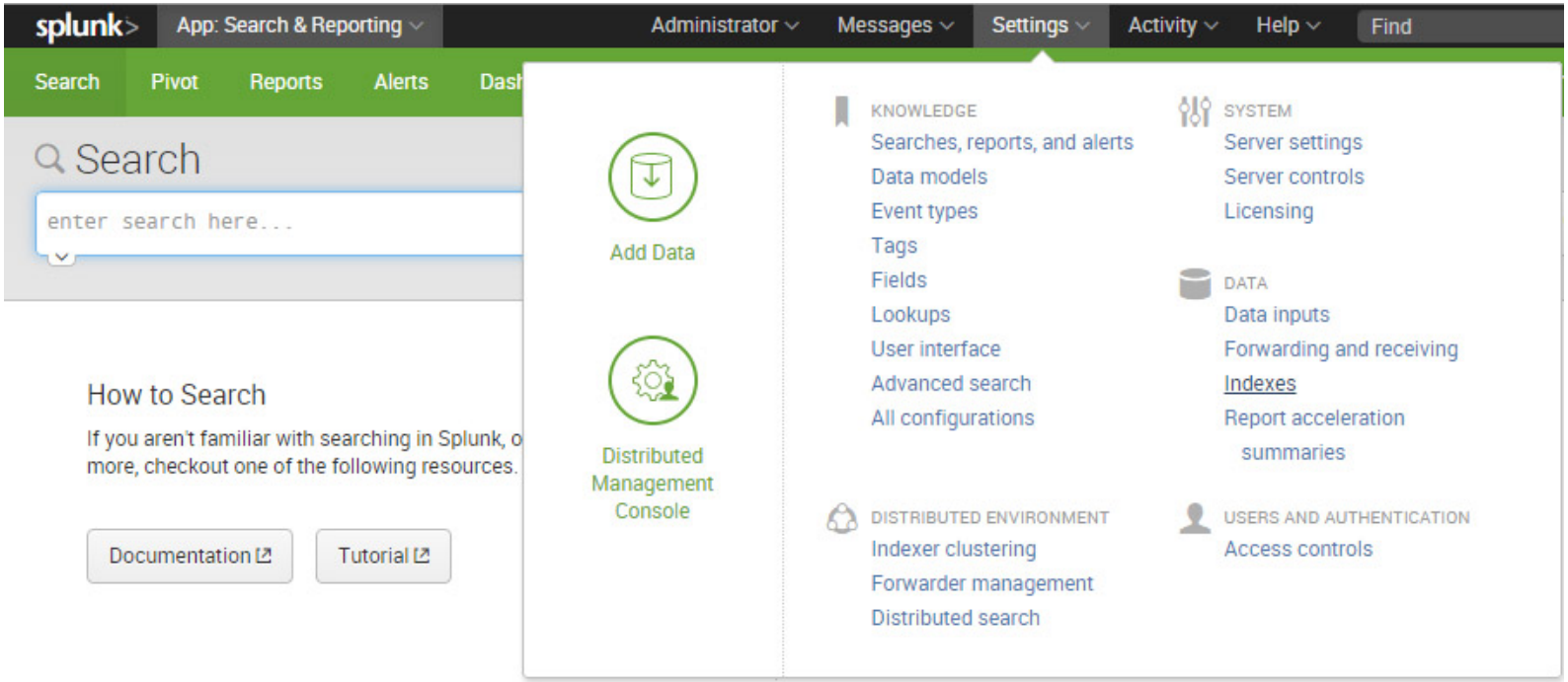
Setting up Automated Alerting on ATO

Building summary index of client logins. Assumptions:

- You have your WEB logs with all the event data indexed in Splunk already. All web events are located within index named: **logs**.
- Field names (or aliases):
  - HTTP request method (GET, POST, HEAD, etc..): **method**
  - URL of page accessed: **page**
  - Username field: **username**
  - IP Address of visitor: **ip**
  - Browser USER_AGENT value: **ua**

Setting up Automated Alerting on ATO

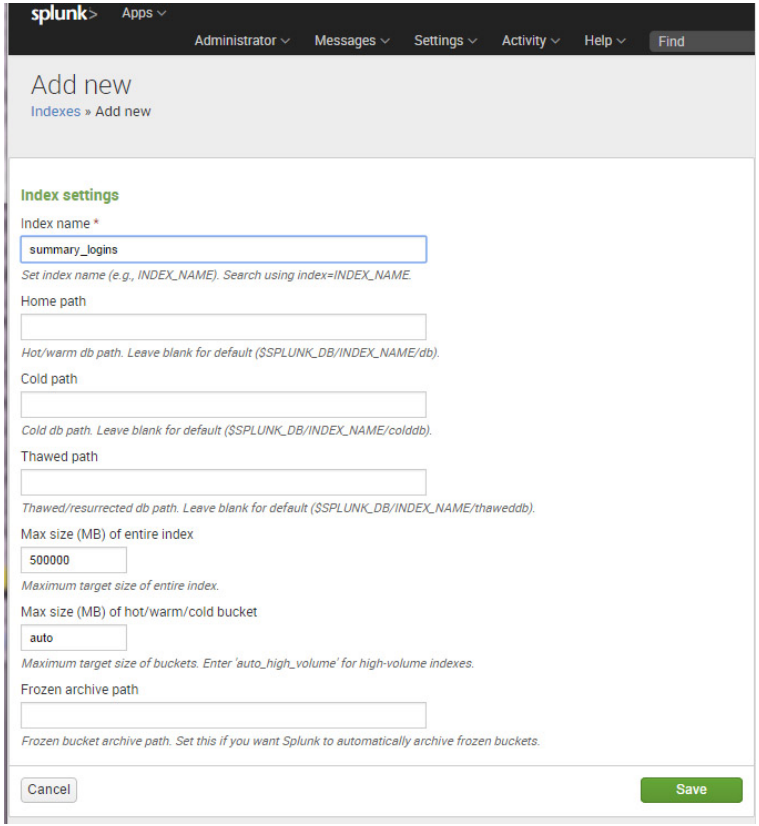
Creating summary index:



Setting up Automated Alerting on ATO

Creating summary index (cont'd):

1. Navigate to your Splunk instance
2. Click on: Settings -> Indexes and then click [New] button
3. Type the name of your summary index, such as "summary_logins"
4. Press [Save] button
5. Next: need to create scheduled summarizing search to push data into summary index at regular intervals
6. Run *fill_summary_index.py* script to backfill summary index with previous data

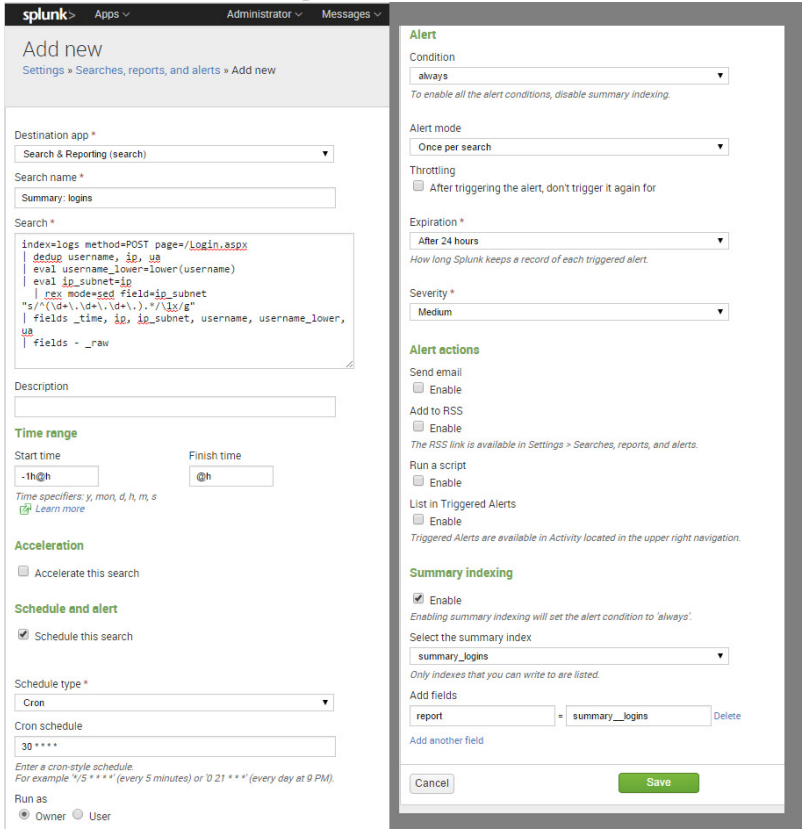


Setting up Automated Alerting on ATO

Creating summarizing search:

1. Navigate to your Splunk instance
2. Click on: navigate to: Settings -> Searches, Reports and Alerts
3. Click [New]
4. Fill the rest according to this image
5. Press [Save]
6. Note: Actual search query is below:

```
index=logs method=POST page=/Login.aspx
| eval username_lower=lower(username)
| dedup username_lower, ip, ua
| eval ip_subnet=ip
| rex mode=sed field=ip_subnet "s/^(\d+\.\d+\.\d+\.).*/\1x/g"
| fields _time, ip, ip_subnet, username, username_lower, ua
| fields - _raw
```



Setting up Automated Alerting on ATO

Summarizing search/ explanations

```
index=logs method=POST page=/Login.aspx
| eval username_lower=lower(username)
| dedup username_lower, ip, ua
| eval ip_subnet=ip
| rex mode=sed field=ip_subnet "s/^(\d+\.\d+\.\d+\.).*/\1x/g"
| fields _time, ip, ip_subnet, username, username_lower, ua
| fields - _raw
```

Note: we need to backfill summary index with historical events using script:

```
%SPLUNK_HOME%/bin/splunk cmd python
fill_summary_index.py -app your-app-name -name
“Summary: logins” -et -45d -lt -1h@h -dedup true -
j 4 -owner admin -auth admin:PasSw0rD
```

- What it does:
- Uses ***index=logs*** to pull all WEB traffic data. This assumes that indexed data already contains either fields or aliases: username, ip, ua, page and method.
- Considers only login-specific events by this query: ***method=POST page=/Login.aspx***
  This of course needs to be modified using specifics of your application.
- Lowercased username is created because username usually is not case sensitive field and users may type it differently: ***| eval username_lower=lower(username)***
- All hourly login events are deduplicated: ***dedup username_lower, ip, ua***
- ip_subnet field is created. If input IP address looks like this: 12.3.45.67 then ip_subnet will take this value: 12.3.45.x
- Then we specify which fields we want to send to our summary index and exclude original _raw field (which is huge and unnecessary to keep).

Setting up Automated Alerting on ATO

The Final BIG search. Reminder:

1. We need to find set of events: multiple logins originated from the same subnet
2. Run custom search per each found login: search for previous IP/ USER_AGENT matches for username
3. If no historical matches found – add it to results
4. If number of results per subnet exceed threshold (75%) – alert!

Challenges:

- #2 – Need somehow to run very custom search per each event
- #3 – Need to return result only if no matches returned by #2

Setting up Automated Alerting on ATO

The Final BIG search. Advanced Negative Look Behind Query:

- **ADVANCED** – ability to run very customized query per each found event within subsearch.
  This is an upgrade from normal subsearch where outer search just uses simple AND or OR logic on fields returned by subsearch. Traditional subsearch allows for some minor customizations via format parameter. But we want more than that.
- **NEGATIVE** – ability to return result signifying *no results found*. Or: return “no matches” if there are one or more results found. This covers the case of: “search for login event for given ‘username’ where either IP subnet or USER_AGENT (or both) are matching the input”. If *not found* – return some sort of flag showing possibly suspicious account activity for given username/ip/user_agent.
- **LOOK BEHIND** – ability to search historical data – in our case 45 days worth of historical ‘summary_logins’ data for each matching event within subsearch.
- **QUERY** pattern - Splunk allows to return actual query to the outer search from subsearch:

```
main search  [ sub search ... | eval search="very custom search" | fields search] ...
```

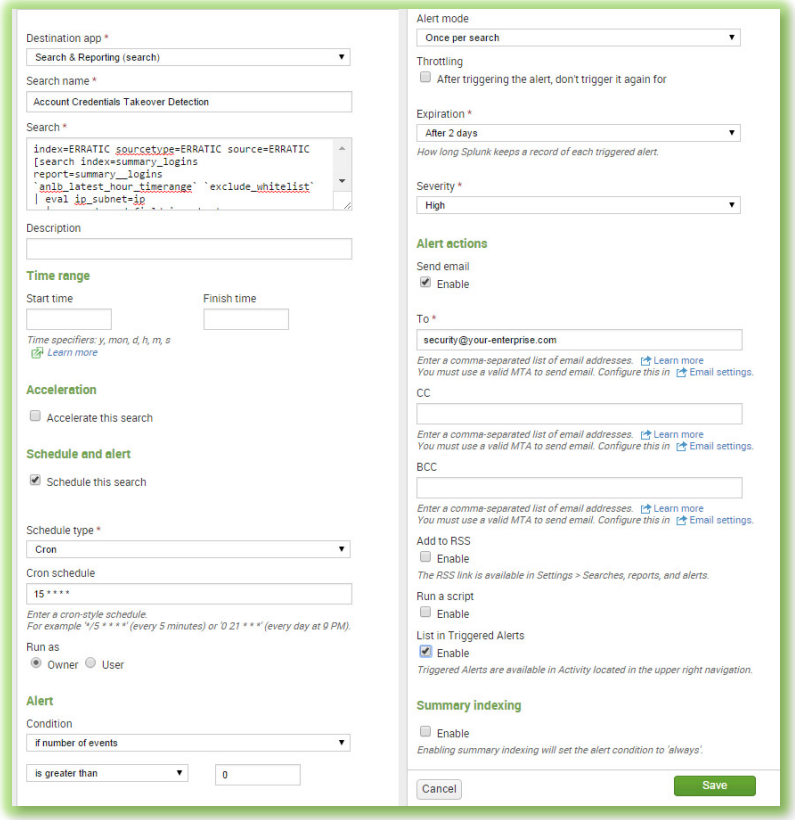
Evolves to:

```
main search   very custom search
```

Setting up Automated Alerting on ATO

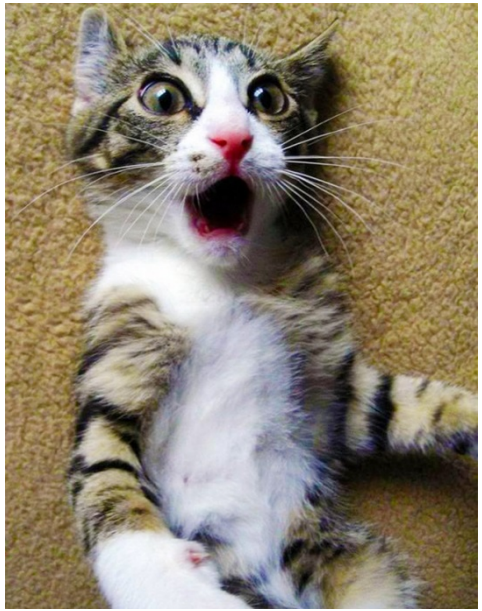
The Final BIG search. Creating main alert:

- Navigate to Settings -> Searches, Reports and Alerts
- Click [New] button
- Fill in the form
- See contents of Search in the next slide…



Setting up Automated Alerting on ATO

The Final BIG search



```
index=ERRATIC sourcetype=ERRATIC source=ERRATIC

[search index=summary_logins report=summary__logins `anlb_latest_hour_timerange` `exclude_whitelist`
| eval ip_subnet=ip
| rex mode=sed field=ip_subnet "s/^(\d+\.\d+\.\d+\.).*/\1x/g"
| dedup ip_subnet, username_lower, ua
| eventstats dc(username_lower) as num_usernames_touched by ip_subnet
| where num_usernames_touched>=`anlb_usernames_touched_threshold`
| fields ip, ip_subnet, username_lower, ua

| eval search_this="
  | append [| stats count AS previous_match_found
    | eval _time=\"" + _time + "\"
    | eval username_lower=\"" + username_lower + "\"
    | eval ip_subnet_orig=\"" + ip_subnet + "\"
    | eval ip_search=\"" + ip + "\"
    | eval ua_search=\"" + ua + "\"
    | appendcols override=1
      [search index=summary_logins report=summary__logins `anlb__lookbehind_timerange`
        username_lower=\"" + username_lower + "\" (ip_subnet=\"" + ip_subnet + "\" OR `anlb_ua_field`=\"" + ua + "\" )
        | head 1
        | eventstats count AS previous_match_found]
    ]
  "

| stats values(search_this) AS all_searches
| eval search=mvjoin(all_searches, " ")
| fields search
]

| eventstats c as total_entries_per_subnet by ip_subnet_orig
| where previous_match_found=0
| eventstats c as unmatched_entries_per_subnet by ip_subnet_orig
| eval percent_unmatched=round(unmatched_entries_per_subnet*100/total_entries_per_subnet, 2)
| where percent_unmatched>=`anlb_percent_unmatched_threshold`
| `iplocation(ip_search)`
| rex mode=sed field=ip_subnet_orig "s/\.x$/.0\/24/g"
| eval show_ratio=tostring(unmatched_entries_per_subnet)+"/"+tostring(total_entries_per_subnet)+" ("+tostring(percent_unmatched)+"%)"
| eval time_string=strftime(_time, "%Y-%m-%d %H:%M:%S")
| sort -unmatched_entries_per_subnet, ip_search, _time
| table ip_subnet_orig, unmatched_entries_per_subnet, show_ratio, Country, Region, City, time_string, ip_search, username_lower, ua_search
| rename
  ip_subnet_orig AS "Suspicious subnet",
  time_string AS "Time",
  ip_search AS "IP",
  username_lower AS "Username",
  ua_search AS "Browser User Agent",
  unmatched_entries_per_subnet AS "Unseen",
  show_ratio AS "Unseen/Total"
```

Setting up Automated Alerting on ATO

The Final BIG search. Macros used

Macros are used to configure “variables” that you can use to tune how alerting query will operate. The following macros are defined:

- `anlb_latest_hour_timerange`:
  `[search index=summary_logins report=summary__logins latest=now | head 1 | eval search="_time>"+tostring(_time-4000)+" _time<="+tostring(_time-0) | fields search]`
  Note: this will allow to “use last hour of available data” vs. “last hour from now”
  Alternatively you may use: `earliest=-1h@h latest=now`
- `anlb__lookbehind_timerange`
  `earliest=-45d@d latest=-1d@d`
- `anlb_percent_unmatched_threshold`
  `75`
- `anlb_ua_field`
  `ua`

Setting up Automated Alerting on ATO

The Final BIG search. Macros used (continued)

- `anlb_usernames_touched_threshold` :
  5
  This is the trigger threshold for the number of accounts simultaneously accessed by the same subnet
- `iplocation(1)`
  ```
  eval Country="Unknown" | iplocation allfields=1 $field_ip$ | eval Country=if(Country="United States", "USA", Country) | eval Location=Region+" - "+City
  ```
  Creates more friendly Country/Region/City fields
- `exclude_whitelist`
  ```
  ip!=123.45.6.78 ip!=234.5.67.0/24
  ```
  This macro defines whitelist to exclude from all searches. In this case I showing you a sample of excluding single IP as well as range of IP addresses defined by CIDR mask

Setting up Automated Alerting on ATO

The Final BIG search. Explanations:

```
index=ERRATIC sourcetype=ERRATIC source=ERRATIC

[search index=summary_logins report=summary__logins `anlb_latest_hour_timerange` `exclude_whitelist`
| eval ip_subnet=ip
| | rex mode=sed field=ip_subnet "s/^(\d+\.\d+\.\d+\.).*/\1x/g"
| dedup ip_subnet, username_lower, ua
| eventstats dc(username_lower) as num_usernames_touched by ip_subnet
| where num_usernames_touched>=`anlb_usernames_touched_threshold`
| fields ip, ip_subnet, username_lower, ua

```

- The Final BIG query consist mostly of big subsearch yielding yet another nested subsearch. There is no main query! Splunk though insist that main query needs to be! Ok, so i fed the monster this “main query”:
  index=ERRATIC sourcetype=ERRATIC source=ERRATIC
  This way Splunk will happily return zero results and get into the subsearch business – where all the magic happens
- This fragment finds all subnets that tried to access multiple accounts within the last hour of *available data*. If found – it returns the set of events for each attempted access with fields: **ip, ip_subnet, username_lower, ua**.

Setting up Automated Alerting on ATO

The Final BIG search. Explanations (continued):

```
| eval search_this="
  | append [|stats count AS previous_match_found
    | eval _time=\"" + _time + "\"
    | eval username_lower=\"" + username_lower + "\"
    | eval ip_subnet_orig=\"" + ip_subnet + "\"
    | eval ip_search=\"" + ip + "\"
    | eval ua_search=\"" + ua + "\"
    | appendcols override=1
      [search index=summary_logins report=summary__logins `anlb__lookbehind_timerange`
        username_lower=\"" + username_lower + "\" (ip_subnet=\"" + ip_subnet + "\" OR `anlb_ua_field`=\"" + ua + "\")
        | head 1
        | eventstats count AS previous_match_found]
      ]
  "
```

The best piece of the whole query! For each event (found at the step previously described) it assembles custom search query to be run against summary index of logins history.
First fragment of this search appends the event with the values of username_lower, ip, ip subnet and user agent (named a bit differently) + very important field: **previous_match_found** with value of 0.
Second piece of search calls appendcols verb with subsearch to be looking into summary index. If anything was found – **previous_match_found** will become 1 – thanks to: … **| head 1** .
Please note that this search is defined and assembled here as pure string.
Later on it will be returned out from the main subsearch to be executed directly.

Setting up Automated Alerting on ATO

The Final BIG search. Explanations (continued):

```
25
26  | stats values(search_this) AS all_searches
27  | eval search=mvjoin(all_searches," ")
28  | fields search
29  ]
30
```

This fragment glues all event-specific searches into one, big-*** search string.
This is what returns out from main subsearch back into main search. If you have lots of matches – the thing returned will be humongous – but nevertheless pretty high performing

Setting up Automated Alerting on ATO

The Final BIG search. Explanations (continued):

```
31  | eventstats c as total_entries_per_subnet by ip_subnet_orig
32  | where previous_match_found=0
33  | eventstats c as unmatched_entries_per_subnet by ip_subnet_orig
34  | eval percent_unmatched=round(unmatched_entries_per_subnet*100/total_entries_per_subnet, 2)
35  | where percent_unmatched>=`anlb_percent_unmatched_threshold`
```

This fragment evaluates value of **previous_match_found** field and checks thresholds that are all defined within macros. We are only interested in suspicious matches exceeding threshold values.

Setting up Automated Alerting on ATO

The Final BIG search. Explanations (continued):

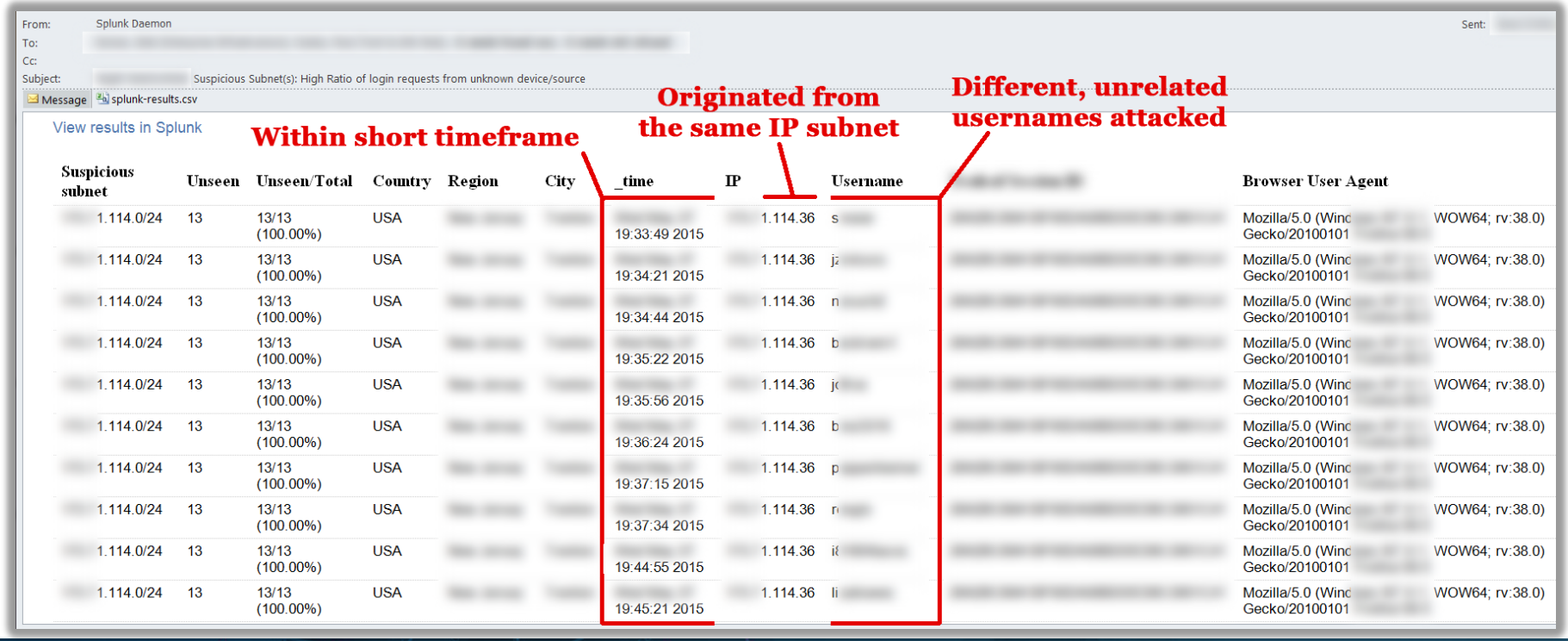
```
36  | `iplocation(ip_search)`
37  | rex mode=sed field=ip_subnet_orig "s/\.x$/.0\/24/g"
38  | eval show_ratio=tostring(unmatched_entries_per_subnet)+"/"+tostring(total_entries_per_subnet)+" ("+tostring(percent_unmatched)+"%)"
39  | eval time_string=strftime(_time, "%Y-%m-%d %H:%M:%S")
40  | sort -unmatched_entries_per_subnet, ip_search, _time
41  | table ip_subnet_orig, unmatched_entries_per_subnet, show_ratio, Country, Region, City, time_string, ip_search, username_lower, ua_search
42  | rename
43    ip_subnet_orig AS "Suspicious subnet",
44    time_string AS "Time",
45    ip_search AS "IP",
46    username_lower AS "Username",
47    ua_search AS "Browser User Agent",
48    unmatched_entries_per_subnet AS "Unseen",
49    show_ratio AS "Unseen/Total"
```

The rest of the search defines few new fields to be returned into the main table, such as ratio of “Unseen/Total” and other explanatory fields that helps to make alert email easily readable.
If suspicious activity is detected – the alert email will be sent to the email addresses specified in alert definition. If nothing bad detected – search will return zero results and no alert will be generated.

Please note that now you may adjust thresholds within the macros without editing main alert definition.

Setting up Automated Alerting on ATO

Detected ATO attack email alert (reminder):



Suspicious subnet	Unseen	Unseen/Total	Country	Region	City	_time	IP	Username	Browser User Agent
1.114.0/24	13	13/13 (100.00%)	USA			19:33:49 2015	1.114.36	s	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:34:21 2015	1.114.36	j	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:34:44 2015	1.114.36	n	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:35:22 2015	1.114.36	b	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:35:56 2015	1.114.36	j	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:36:24 2015	1.114.36	b	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:37:15 2015	1.114.36	p	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:37:34 2015	1.114.36	r	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:44:55 2015	1.114.36	i	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101
1.114.0/24	13	13/13 (100.00%)	USA			19:45:21 2015	1.114.36	li	Mozilla/5.0 (Winc WOW64; rv:38.0) Gecko/20100101

Tuning

Adjust limits.conf to prevent possible subsearch timeouts:

- Add this to: %SPLUNK_HOME%/etc/apps/**your-app**/local/limits.conf

```
[subsearch]
maxtime = 600

[join]
subsearch_maxtime = 600
subsearch_timeout = 800
```

- Average runtime of ATO alerting query: 120-180 seconds.
- Above settings were sufficient to handle traffic of 5,000,000 hits/day and up to 5 new client web sessions **per second**.
- Make sure to adjust values of macros to tune alert triggers thresholds to better match specifics of your business and traffic patterns.

False Positives?

- Add IP ranges of aggregators, large corporate automated systems, etc.. into `` `exclude_whitelist` `` macro.
- Current average rate of false positives: 0-5 alerts **per day**.
- Most (95%) of false positives – when client uses new computer, at a new location and forgot his username (trying multiple variations of his own username).

Recap of 5 Take-Aways

- Solution described was able to detect and alert on bank account takeover attempts better and faster than any other security tools
- In a number of cases – Splunk ATO alert was **the only tool** that detected attempted fraud
- Splunk right out of the box can easily be turned into efficient and flexible fraud detection and security analytics framework
- Splunk can generate and execute it's own searches very efficiently
- The more data – the better

Full description of solution + source code:

http://www.mensk.com/ato1

Detecting More Sophisticated Fraudsters

Bonus Material

Above method detects 65-90% of ATO attacks

What it does not detect:

- Attacks spread across different IP subnets. Launched via botnets
- Attacks initiated on a single user/client account
- Attacks initiated from victim's IP or subnet:
  disgruntled employee, family member, caregiver, malware

Detecting More Sophisticated Fraudsters

Bonus Material (continued)

Solution? Cumulative behavior risk scoring

1. **Calculate baseline of typical customer session behavior and assign risk score to anomalies**
2. **Define customer session activities with above zero risk and assign risk scores to risky actions as well as to digital forensic data:**
   - Session hits than may cause money movements, profile and password changes, securities trading, unusual/unseen IP address and browser User Agent. Add risk scores to timing between login and risky actions, as well as to order or actions
3. **Calculate session risk score and alert on broken thresholds**

Detecting More Sophisticated Fraudsters

Bonus Material (continued)

Actual implementation steps:

1. Generated summary index of all user sessions for reference with past
2. Generated secondary summary index of session metrics for each 4-hour window (total average hits per session, total average session duration)
3. Created scheduled search / alert to calculate cumulative risk score of each session within 1 hour sliding window

Detecting More Sophisticated Fraudsters

Bonus Material (continued)

Actual implementation steps: SPL template:

```
index=logs .. | transaction session_id ..

| eval riskscore=0 | eval riskmsg=0 ..
| eval test=mvfind(xpages, "(?i)/updatepassword") | eval addscore=20
  | eval riskscore=if(isnull(test), riskscore, riskscore+addscore)
  | eval riskmsg=if(isnull(test), riskmsg, riskmsg+"|(+"+addscore+") Password update detected!")

| ..test for more risky actions to add to total session risk score..

| where riskscore>=95
| table _time, ip, riskscore, riskmsg, ..
```

Detecting More Sophisticated Fraudsters

Bonus Material (continued)

Actual implementation steps,
Detailed description + source code:

http://www.mensk.com/ato2

Credits and Inspirations:

- **Mica Roth-Martin** – Splunk Global Accounts Manager
- **Jeff Champagne** – Splunk Professional Services Team
- **Rob Perdue** – 8020 Labs, .Conf2014 presenter / Detecting Fraud with Risk Scoring

splunk>

THANK YOU

Gleb Esman

Blog, source code and updates: **http://www.mensk.com/**
LinkedIn: **https://www.linkedin.com/in/glebesman/**

splunk>

Questions?

Note: Birds of Feather sessions for security and fraud discussions:

Thursday, Sep 24th: 11:00-14:00