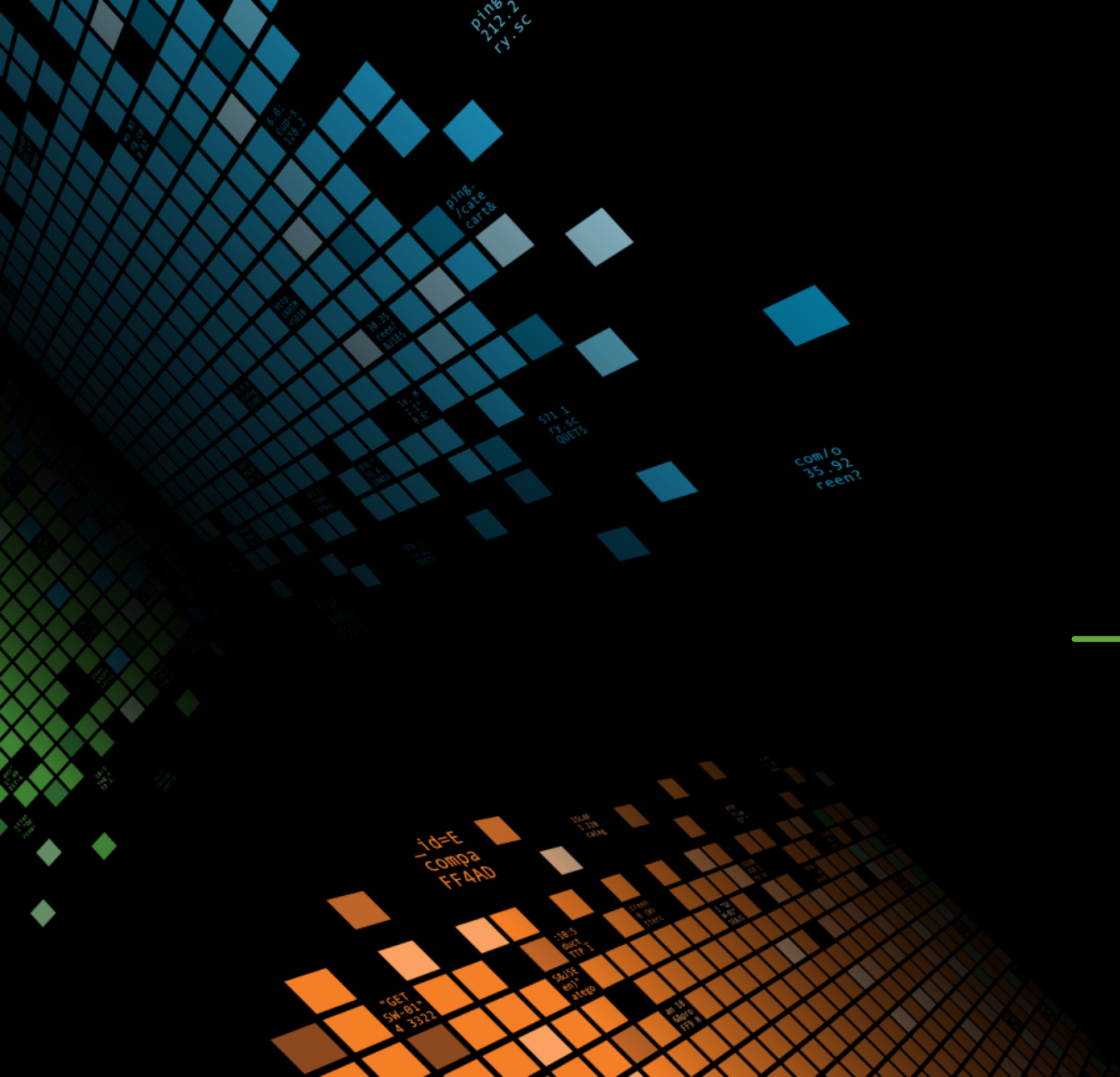
splunk> .conf2017

Beyond “Regular” Regular Expressions

Cary Petterborg | Splunk Architect | LDS Church

August 8, 2017



Boilerplate

splunk> .conf2017

Forward-Looking Statements

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.

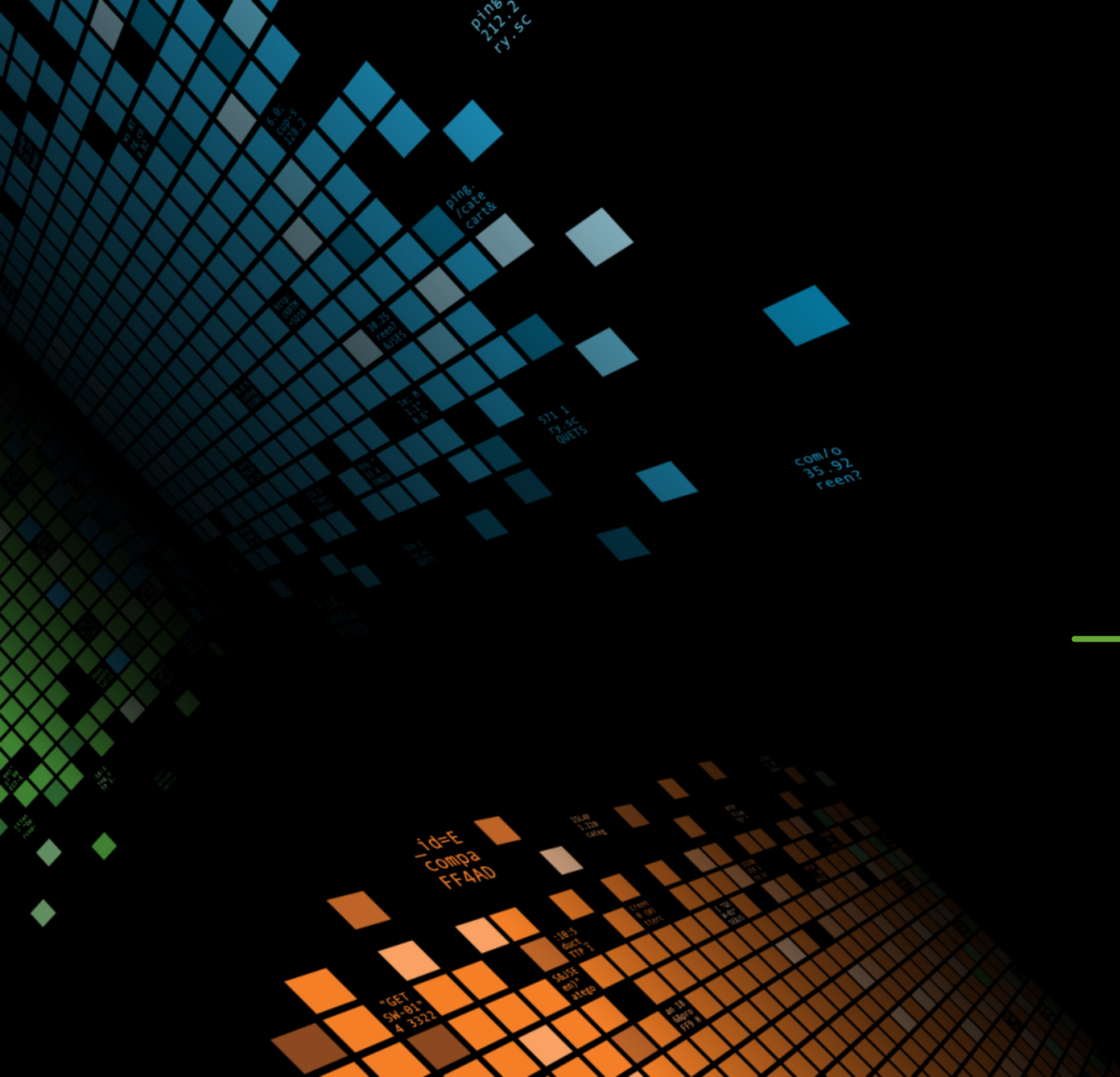
The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward looking statements 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.

splunk> .conf2017

My Disclaimer

During the course of this presentation, I may make references to my employer, The Church of Jesus Christ of Latter-day Saints. This should not be taken as an endorsement of Splunk or Splunk products by the LDS Church.

splunk> .conf2017

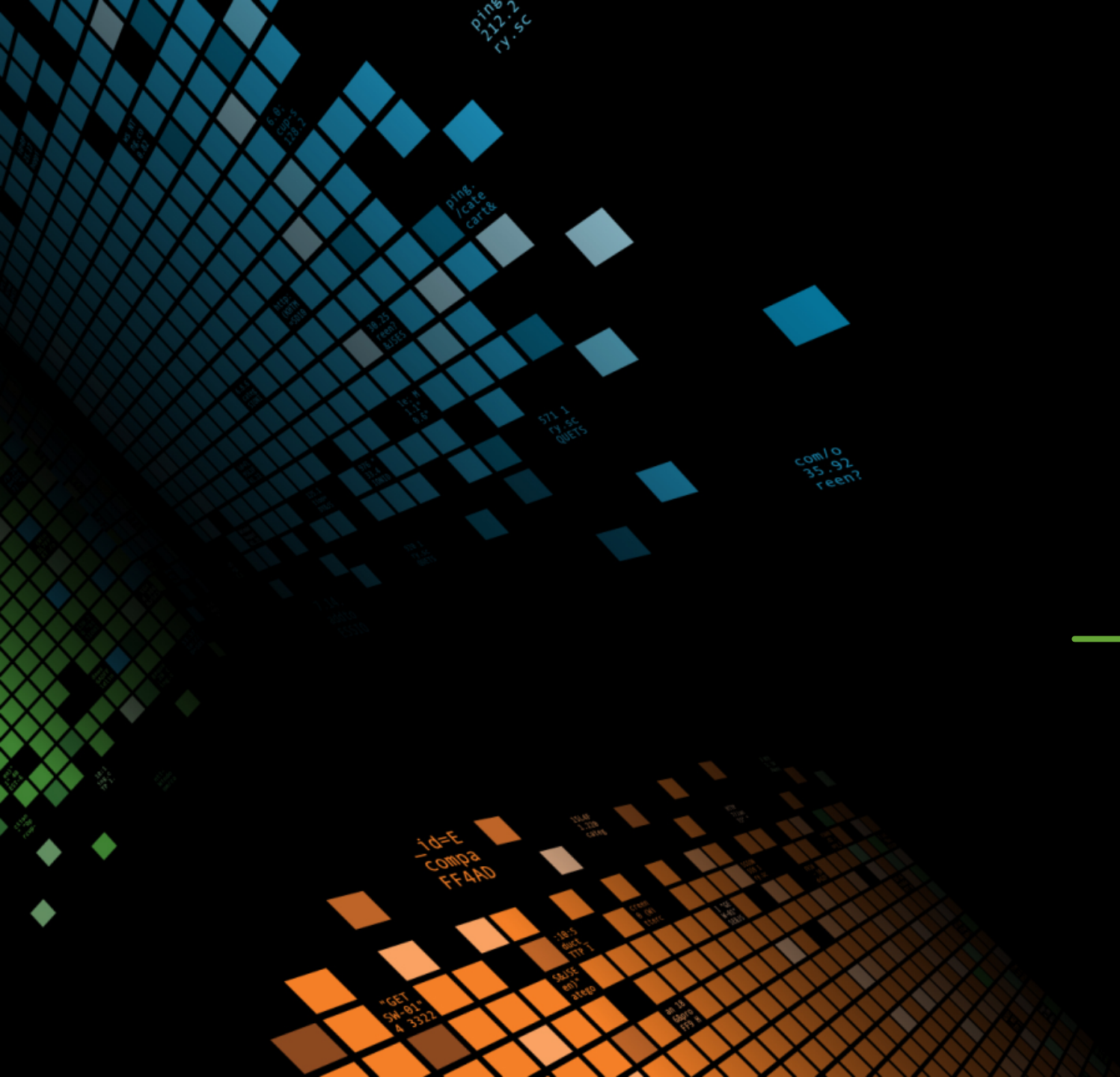


About Me

Who is Cary Petterborg

- Splunk user and administrator for 5.5 years
- Monitoring Engineer for 10 years
- Web developer for 23 years
- Software engineer for 37 years
- Many languages from assembly to Ruby
- Application development including Flight Sim, DB systems, and Web
- Works for the LDS Church in Salt Lake City
- Speaker at .conf 201[567]
- SplunkTrust Member 2018

splunk> .conf2017



Purpose

My purpose here today is to…

- Help you **control your data** instead of letting your data control you
- Regular expressions give you that control

Where's Waldo?

Can you easily pick out all the female Waldo's?

Picture courtesy of Albanpix.com

Regular Expressions help you…

- Find the distinctions within similar data
- Isolate the value properties from the noise
- Find the needles in the haystacks
- Break data into usable, constituent parts

Why Do I Like Regular Expressions?

- Using Regular Expressions since the mid 80's
- Started using regex with lex/yacc/sed/grep for software development
- Realized the power of regex quickly
- Taught classes on regex
- Love working with regex stuff in Splunk and other utilities
- Regex is an important skill, and I want to share my knowledge
- **Have Rex – Will Conquer**

One day, you too…

REGULAR EXPRESSION

1 MATCH - 9799 STEPS

```
/ ^\w{3}\s\d\d \d\d:\d\d:\d\d (?P<host>\w+).*Description=\s*(?P<descr>[\s\S]+20\d\d):[\s\S]*(MediaResourceList
Name\s+:\s+(?P<media_res_list_name>\w+)[\s\S]*MediaResourceType\s+:\s+(?P<media_res_type>\d+)|RouteListName\
s+:\s+(?P<route_list_name>[^,;]+)[,;\s]*Reason=(?P<reason>\d+)[,;]\s*RouteGroups\((?P<route_grps>[^\)]+)\)|T
imeStamp\s+:\s+(?P<timestamp>\d+\/\d+\/\d+\s+\d+:\d+\s+\w+)\s+LoginFrom\s+:\s+(?P<login_from>[.\d]+)\s+Inter
face\s+:\s+(?P<interface>\w+)\s+UserID\s+:\s+(?P<user_id>\w+))[\s\S]+AppID\s+:\s+(?P<app_id>[\w\s]+)[\s\S]*C
lusterID\s+:\s+(?P<cluster_id>[\w-]*)[\s\S]+NodeID\s+:\s+(?P<node_id>\w+)[\s\S]+TimeStamp\s+:\s+(?P<ts>\w{3}
\s\w{3}\s+\d+\s+\d+:\d+:\d+\s+\w+\s+\d+)[\s\S]*::Status=(?P<status>\d+,[a-z]+)\^Severity=(?P<severity>\w+)\^
Acknowledged=(?P<acked>\w+)\^CUSTOMER=(?P<customer>[^\^]+)\^Private IP Address=(?P<priv_ip>[^\^]+)\^Default
Alarm Name=(?P<def_alarm_name>[^\^]+)\^Managed Object=(?P<object>[^\^]+)\^Managed Object Type=(?P<obj_type>[
^\^]+)\^MODE=(?P<mode>[^;]+);Alarm ID=(?P<alarm_id>\d+)\^Component=(?P<component>[^\\]+)\\x00000 / gmixXsuUAJ
```

TEST STRING

```
Jan 14 11:18:46 h32467 CPCMh32467: %local7-2-ALARM: 16$Description=  Number of MediaResourceListExhausted events exceeds confi
gured threshold during configured interval 5 within 60 minutes  on cluster US-UT-DIR-Production.  There are 6 MediaResourceLis
tExhausted events (up to 30) received during the monitoring interval From Thu Jan 14 10:50:52 MST 2016 to Thu Jan 14 11:50:52
MST 2016:   MediaResourceListName : US-UT-DIR-Campus-NoVideo MediaResourceType : 2 AppID : Cisco CallManager ClusterID : US-UT
-DIR-Production NodeID : dirvp211a  TimeStamp : Thu Jan 14 11:17:53 MST 201::Status=1,active^Severity=minor^Acknowledged=no^CU
STOMER=Cisco Prime Collaboration^Private IP Address=10.308.5.22^Default Alarm Name=MediaListExhausted^Managed Object=10.308.5.
22^Managed Object Type=Communications Manager^MODE=2;Alarm ID=659201239^Component=dirvp212a.wh.hamsterfarm.net\x00000
```

One day, you too…

Regular Expression

Regular Expression Reference | View in Search

```
^\w{3}\s\d\d \d\d:\d\d:\d\d (?P<host>\w+).*Description=\s*(?P<descr>[\s\S]+20\d\d):[\s\S]*(MediaResourceListName\s+:\s+(?P<media_res_list_name>\w+)[\s\S]*MediaResourceType\s+:\s+(?P<media_res_type>\d+)|RouteListName\s+:\s+(?P<route_list_name>[^,]+)[,;]\s*Reason=(?P<reason>\d+)[,;]\s*RouteGroups\((?P<route_grps>[^\)]+)\)|TimeStamp\s+:\s+(?P<timestamp>\d+\/\d+\/\d+\s+\d+:\d+\s+\w+)\s+LoginFrom\s+:\s+(?P<login_from>[.\d]+)\s+Interface\s+:\s+(?P<interface>\w+)\s+UserID\s+:\s+(?P<user_id>\w+)|[\s\S]+AppID\s+:\s+(?P<app_id>[\w\s]+)[\s\S]*ClusterID\s+:\s+(?P<cluster_id>[\w-]*)[\s\S]+NodeID\s+:\s+(?P<node_id>\w+)[\s\S]+TimeStamp\s+:\s+(?P<ts>\w{3}\s\w{3}\s+\d+\s+\d+:\d+:\d+\s+\w+\s+\d+)[\s\S]*::Status=(?P<status>\d+,[a-z]+)\^Severity=(?P<severity>\w+)\^Acknowledged=(?P<acked>\w+)\^CUSTOMER=(?P<customer>[^\^]+)\^Private IP Address=(?P<priv_ip>[^\^]+)\^Default Alarm Name=(?P<def_alarm_name>[^\^]+)\^Managed Object=(?P<object>[^\^]+)\^Managed Object Type=(?P<obj_type>[^\^]+)\^MODE=(?P<mode>[^;]+);Alarm ID=(?P<alarm_id>\d+)\^Component=(?P<component>[^\\]+)\\x00000
```

Preview | Save

Events | host | descr | media_res_list_name | media_res_type | route_list_name | reason | route_grps | timestamp | login_from | interface | user_id | app_id | cluster_id | node_id | ts | status | severity | acked

✓ 10 events (before 7/11/16 4:29:50.000 PM)

Original search included: ? ☑ 20 per page

filter | Apply | Sample: 1,000 events | All events | All Events | Matches | Non-Matches

_raw	host	descr
✓ Jan 14 11:18:46 h32467 CPCMh32467: %local7-2-ALARM: 16$Description= Number of MediaResourceListExhausted events exceeds configured threshold during configured interval 5 within 60 minutes on cluster US-UT-DIR-Production. There are 6 MediaResourceListExhausted events (up to 30) received during the monitoring interval From Thu Jan 14 10:50:52 MST 2016 to Thu Jan 14 11:50:52 MST 2016 : MediaResourceListName : US-UT-DIR-Campus-NoVideo MediaResourceType : 2 AppID : Cisco CallManager ClusterID : US-UT-DIR-Production NodeID : dirvp211a TimeStamp : Thu Jan 14 11:17:53 MST 201::Status=1,active ^Severity=minor ^Acknowledged=no ^CUSTOMER=Cisco Prime Collaboration ^Private IP Address=10.308.5.22 ^Default Alarm Name=MediaListExhausted ^Managed Object=10.308.5.22 ^Managed Object Type=Communications Manager ^MODE=2 ;Alarm ID=659201239 ^Component=dirvp212a.wh.hamsterfarm.net \x00000	h32467	Number of MediaResourceListExhausted events exceeds configured threshold during configured interval 5 within 60 minutes on cluster US-UT-DIR-Production. There are 6 MediaResourc
✓ Jan 14 10:58:14 h32467 CPCMh32467: %local7-2-ALARM: 16$Description= Number of RouteListExhausted events exceeds configured threshold during configured interval 0 within 60 minutes on cluster BRVP. There are 1 RouteListExhausted events (up to 30) received during the monitoring interval From Thu Jan 14 15:48:46 BRST 2016 to Thu Jan 14 16:48:46 BRST 2016 : RouteListName : Local Route Group No Translations ; Reason=41 ; RouteGroups(BR-SP-Brazil-TPL-AO) AppID : Cisco CallManager ClusterID : BRVP NodeID : BRVP2 TimeStamp : Thu Jan 14 15:54:45 BRST 2016 ::Status=1,active ^Severity=minor ^Acknowledged=no ^CUSTOMER=Cisco Prime Collaboration ^Private IP Address=10.555.41.20 ^Default Alarm Name=RouteListExhausted ^Managed Object=10.555.41.20 ^Managed Object Type=Communications Manager ^MODE=2 ;Alarm ID=659617997 ^Component=BRVP1.heavensgate.org \x00000	h32467	Number of RouteListExhausted events exceeds configured threshold during configured interval 0 within 60 minutes on cluster BRVP. There are 1 RouteListExhausted events (up to 30) re
✓ Jan 14 10:53:27 h32467 CPCMh32467: %local7-2-ALARM: 16$Description=Number of MediaResourceListExhausted events exceeds configured threshold during configured interval 5 within 60 minutes\x00012 on cluster US-UT-DIR-Production.\x00012\x00012There are 6 MediaResourceListExhausted events (up to 30) received during the monitoring interval From Thu Jan 14 09:50:51 MST 2016 to Thu Jan 14 10:50:51 MST 2016 : \x00012\x00012MediaResourceListName : NULL_LIST\x00012MediaResourceType : 1\x00012AppID : Cisco CallManager\x00012ClusterID : US-UT-DIR-Production\x00012NodeID : dirvp211b\x00012 TimeStamp : Thu Jan 14 10:13:44 MST 2016\x00012\x00012MediaResourceL::Status=2,cleared ^Severity=minor ^Acknowledged=no ^CUSTOMER=Cisco Prime Collaboration ^Private IP Address=10.308.5.22 ^Default Alarm Name=MediaListExhausted ^Managed Object=10.308.5.22 ^Managed Object Type=Communications Manager ^MODE=2 ;Alarm ID=659201239 ^Component=dirvp212a.wh.hamsterfarm.net \x00000	h32467	Number of MediaResourceListExhausted events exceeds configured threshold during configured interval 5 within 60 minutes\x00012 on cluster US-UT-DIR-Production.\x00012\x00012T
✓ Jan 14 10:53:06 h32467 CPCMh32467: %local7-2-ALARM: 16$Description= Number of MediaResourceListExhausted events exceeds configured threshold during configured interval 0 within 60 minutes on cluster ARVP. There are 1 MediaResourceListExhausted events (up to 30) received during the monitoring interval From Thu Jan 14 14:48:15 ART 2016 to Thu Jan 14 15:48:15 ART 2016 : MediaResourceListName : UY-MO-MNTVD-TPLCMPLX MediaResourceType : 7 AppID : Cisco CallManager ClusterID : ARVP NodeID : ARVP2 TimeStamp : Thu Jan 14 14:51:15 ART 2016 ::Status=1,active ^Severity=minor ^Acknowledged=no ^CUSTOMER=Cisco Prime Collaboration ^Private IP Address=10.555.72.21 ^Default Alarm Name=MediaListExhausted ^Managed Object=10.555.72.21 ^Managed Object Type=Communications Manager ^MODE=2 ;Alarm ID=659600865 ^Component=ARVP2.LDSChurch.org \x00000	h32467	Number of MediaResourceListExhausted events exceeds configured threshold during configured interval 0 within 60 minutes on cluster ARVP. There are 1 MediaResourceListExhausted
✓ Jan 14 10:28:15 h32467 CPCMh32467: %local7-2-ALARM: 16$Description=Number of RouteListExhausted events exceeds configured threshold during configured interval 0 within 60 minutes\x00012 on cluster DE-HE.\x00012\x00012There are 1 RouteListExhausted events (up to 30) received during the monitoring interval From Thu Jan 14 17:22:58 CET 2016 to Thu Jan 14 18:22:58 CET 2016 : \x00012\x00012RouteListName : LocalRouteGroupOnly, Reason=41, RouteGroups(FR-J-Torcy-SC)\x00012AppID : Cisco CallManager\x00012ClusterID : DE-HE\x00012NodeID : eadebhgscvp002\x00012 TimeStamp : Thu Jan 14 17:38:52 CET 2016\x00012\x00012::Status=2,cleared ^Severity=minor ^Acknowledged=no ^CUSTOMER=Cisco Prime Collaboration ^Private IP Address=10.277.4.21 ^Default Alarm Name=RouteListExhausted ^Managed Object=10.277.4.21 ^Managed Object Type=Communications Manager ^MODE=2 ;Alarm ID=658775520 ^Component=eadebhgscvp002.heavensgate.org \x00000	h32467	Number of RouteListExhausted events exceeds configured threshold during configured interval 0 within 60 minutes\x00012 on cluster DE-HE.\x00012\x00012There are 1 RouteListExha

Splunk and Regular Expressions

splunk> .conf2017

Where do you use regular expression in Splunk?

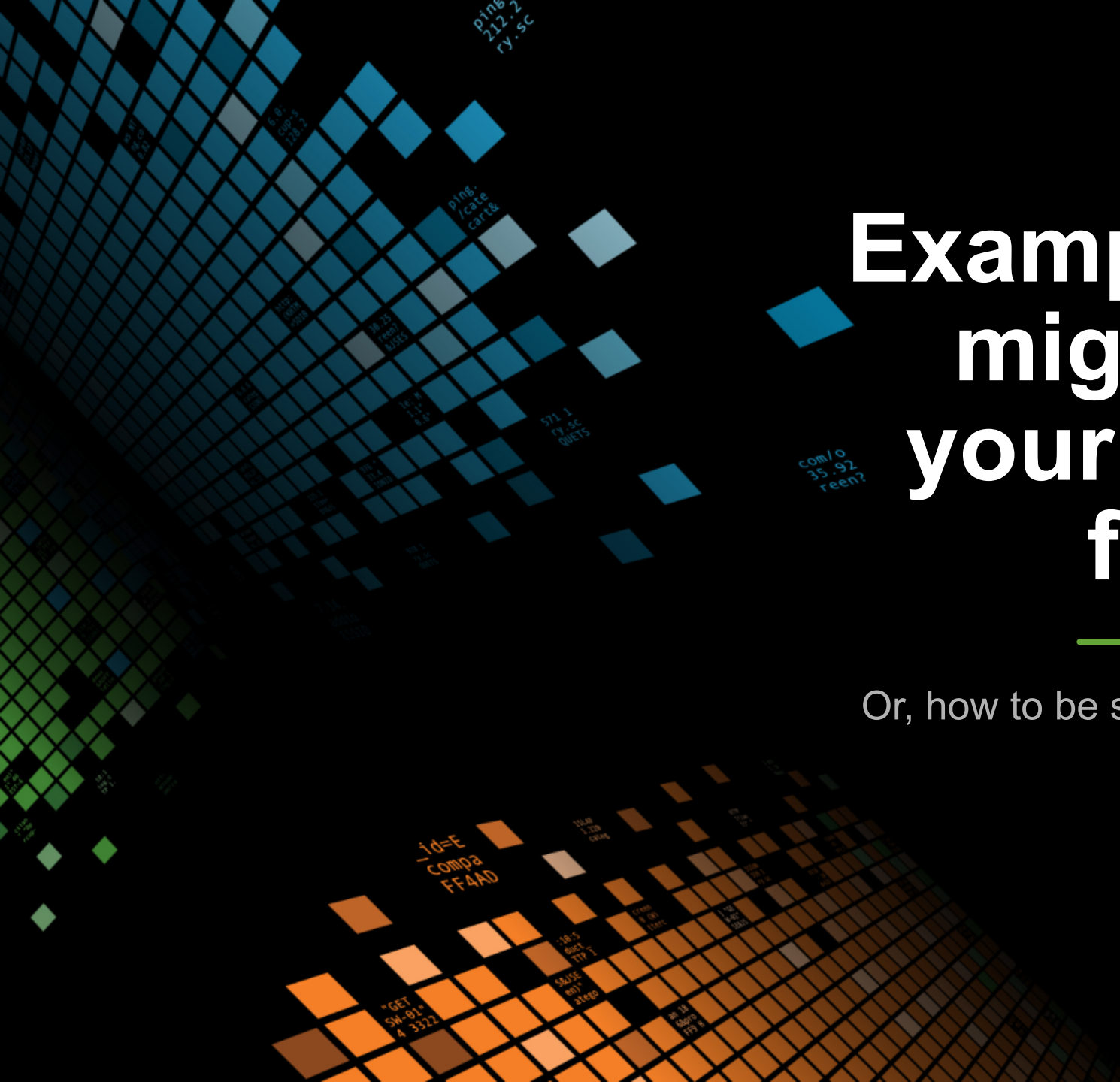
- Field extractions
- The **rex** and **regex** search commands
- In *props.conf, transforms.conf* and other *.conf* files
- Data feeds (probably external to Splunk itself)
- Note: Splunk regular expressions are PCRE (Perl Compatible Regular Expressions) and use the PCRE C library.

Splunk Field Extraction Tool

splunk> .conf2017

Field Extraction Tool

- GUI tool in the web UI of Splunk
- Simple to use, and you can visually see the results of the regex on the events
- Pretty good *for a start*, but not *always* good for a final result
- Not able to optimize or do anything complex
- Makes mistakes if you don’t have regular data



Example of why you might want to use your own regex for field extraction

Or, how to be smarter than the Field Extraction Tool

splunk> .conf2017

Multi-format Security Data in…

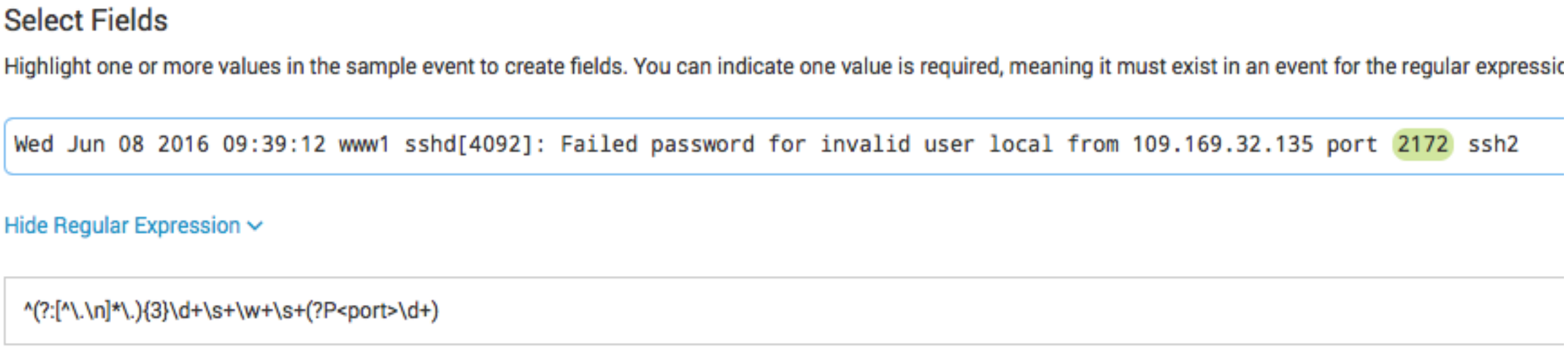
secure.log

```
Mon Jun 06 2016 02:29:19 www1 sshd[3849]: Failed password for root from 128.241.220.82 port 2253 ssh2
Mon Jun 06 2016 02:29:24 www1 sshd[4267]: Failed password for invalid user administrator from 128.241.220.82 port 1715 ssh2
Mon Jun 06 2016 02:29:44 www1 sshd[5001]: Failed password for invalid user george from 128.241.220.82 port 2212 ssh2
Mon Jun 06 2016 02:30:13 www1 sshd[1638]: pam_unix(sshd:session): session opened for user djohnson by (uid=0)
Mon Jun 06 2016 02:30:13 www1 sshd[1480]: Failed password for invalid user yp from 128.241.220.82 port 2808 ssh2
Mon Jun 06 2016 02:30:22 www1 sshd[2291]: Failed password for invalid user email from 128.241.220.82 port 4995 ssh2
Mon Jun 06 2016 02:30:26 www1 sshd[4761]: Failed password for invalid user local from 128.241.220.82 port 1271 ssh2
Mon Jun 06 2016 02:30:50 www1 sshd[4986]: Failed password for invalid user mysql from 128.241.220.82 port 2076 ssh2
Mon Jun 06 2016 02:31:19 www1 sshd[81145]: pam_unix(sshd:session): session opened for user djohnson by (uid=0)
Mon Jun 06 2016 02:31:19 www1 sshd[2021]: Failed password for myuan from 10.1.10.172 port 4468 ssh2
Mon Jun 06 2016 02:31:28 www1 sshd[1155]: Failed password for invalid user yp from 10.1.10.172 port 1822 ssh2
Mon Jun 06 2016 02:31:32 www1 sshd[1632]: Failed password for invalid user elena_andubasquet from 10.1.10.172 port 2074 ssh2
Mon Jun 06 2016 02:31:54 www1 sshd[4333]: Failed password for root from 10.1.10.172 port 2772 ssh2
Mon Jun 06 2016 02:32:00 www1 sudo: djohnson : TTY=pts/0 ; PWD=/home/djohnson ; USER=root ; COMMAND=/bin/su
Mon Jun 06 2016 02:32:00 www1 sshd[3697]: Failed password for invalid user whois from 10.1.10.172 port 1246 ssh2
Mon Jun 06 2016 02:32:19 www1 sshd[5985]: Failed password for invalid user testuser from 10.1.10.172 port 2597 ssh2
```

splunk> .conf2017

Simple FET extraction of Port

This is what the FET gives you:



Intelligent extraction of Port

Regular Expression

```
port\s+(?P<port>\d+)
```

Notes on Named Capture Groups

- (?P<name>…) === (?<name>...)
- The **P** is optional (came from **P**ython), but it is usually considered more correct
- Splunk FET will use (?P<name>...), so why not make things similar?

BUT

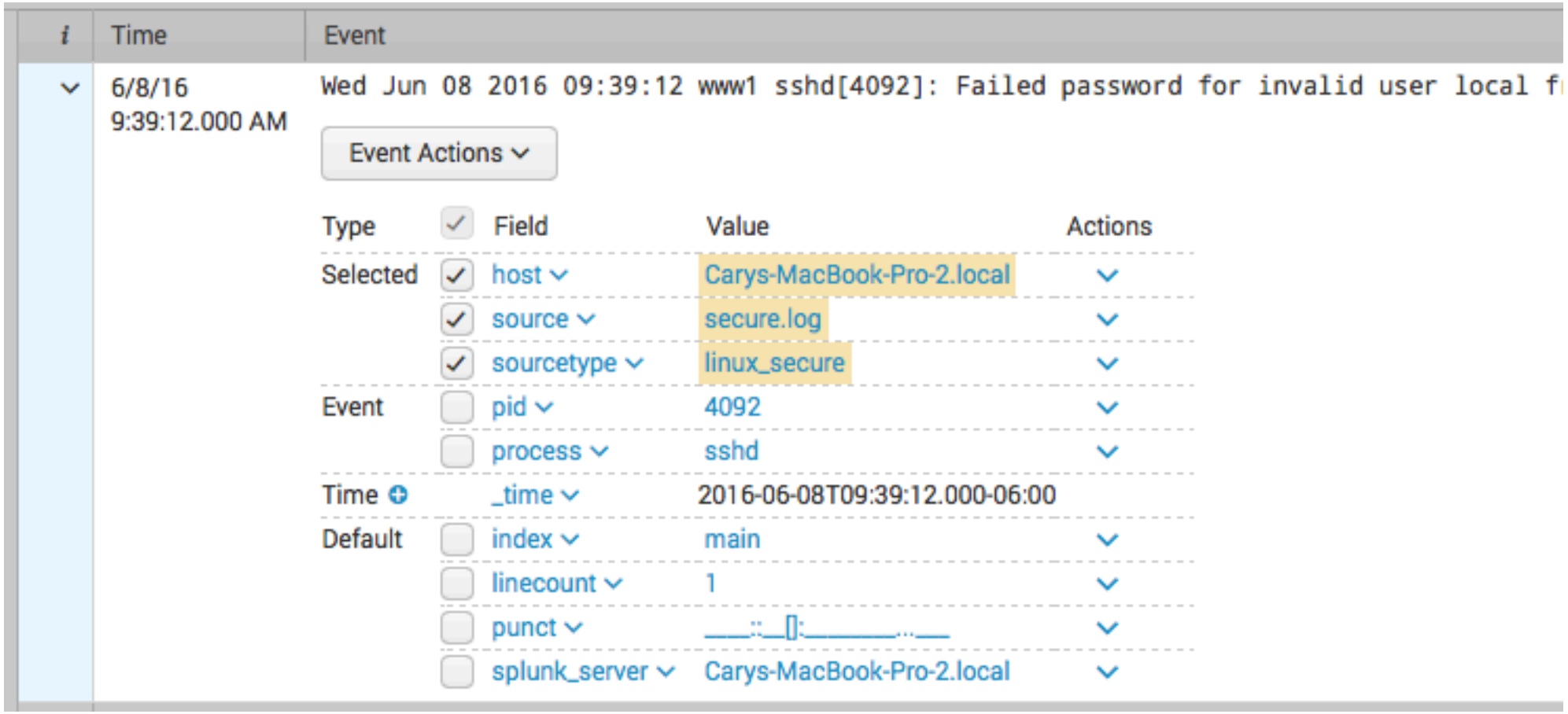
- Do it the way you feel most comfortable

Goal: *user* from all entries using one regex

```
1 sshd[3697]: Failed password for invalid user whois from 10.1.10.172 port 124
1 sudo: djohnson ;  TTY=pts/0 ; PWD=/home/djohnson ; USER=root ; COMMAND=/bin/
1 sshd[4333]: Failed password for root from 10.1.10.172 port 2772 ssh2
1 sshd[1632]: Failed password for invalid user elena_andubasquet from 10.1.10.
1 sshd[1155]: Failed password for invalid user yp from 10.1.10.172 port 1822 s
1 sshd[2021]: Failed password for myuan from 10.1.10.172 port 4468 ssh2
1 sshd[81145]: pam_unix(sshd:session): session opened for user djohnson by (ui
1 sshd[4986]: Failed password for invalid user mysql from 128.241.220.82 port
1 sshd[4761]: Failed password for invalid user local from 128.241.220.82 port
```

splunk> .conf2017

There is no user automatically extracted



FET Failed

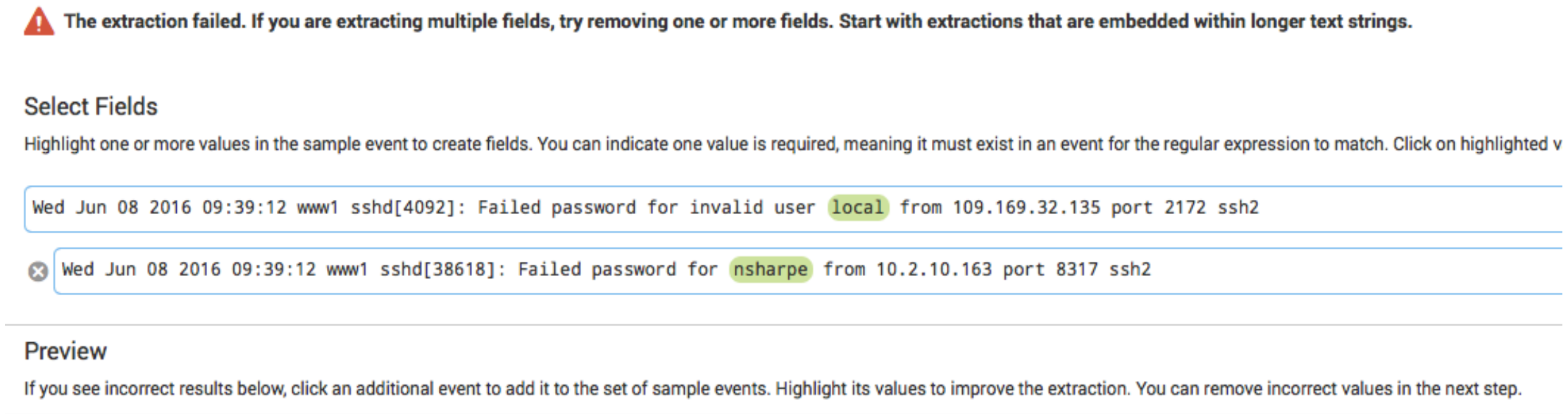
Gets wrong values from some events

```
: Failed password for invalid user local from 109.169.32.135 port 2
]: Failed password for nsharpe from 10.2.10.163 port 8317 ssh2
: Failed password for invalid user dean from 109.169.32.135 port 38
: Failed password for invalid user operator from 109.169.32.135 por
: Failed password for invalid user itmadmin from 109.169.32.135 por
: Failed password for mail from 84.34.159.23 port 1190 ssh2
: Failed password for sync from 84.34.159.23 port 2530 ssh2
: Failed password for invalid user inet from 10.3.10.46 port 1516 s
]: pam_unix(sshd:session): session closed for user myuan by (uid=0)
: Failed password for invalid user administrator from 10.3.10.46 po
```

splunk> .conf2017

FET Failed After More Lines Added

Trying to add an additional line and extracting *user* doesn't work



splunk> .conf2017

Let’s look at the generated REGEX

Not pretty, not easy to change/fix, not efficient. So, let’s fix it.

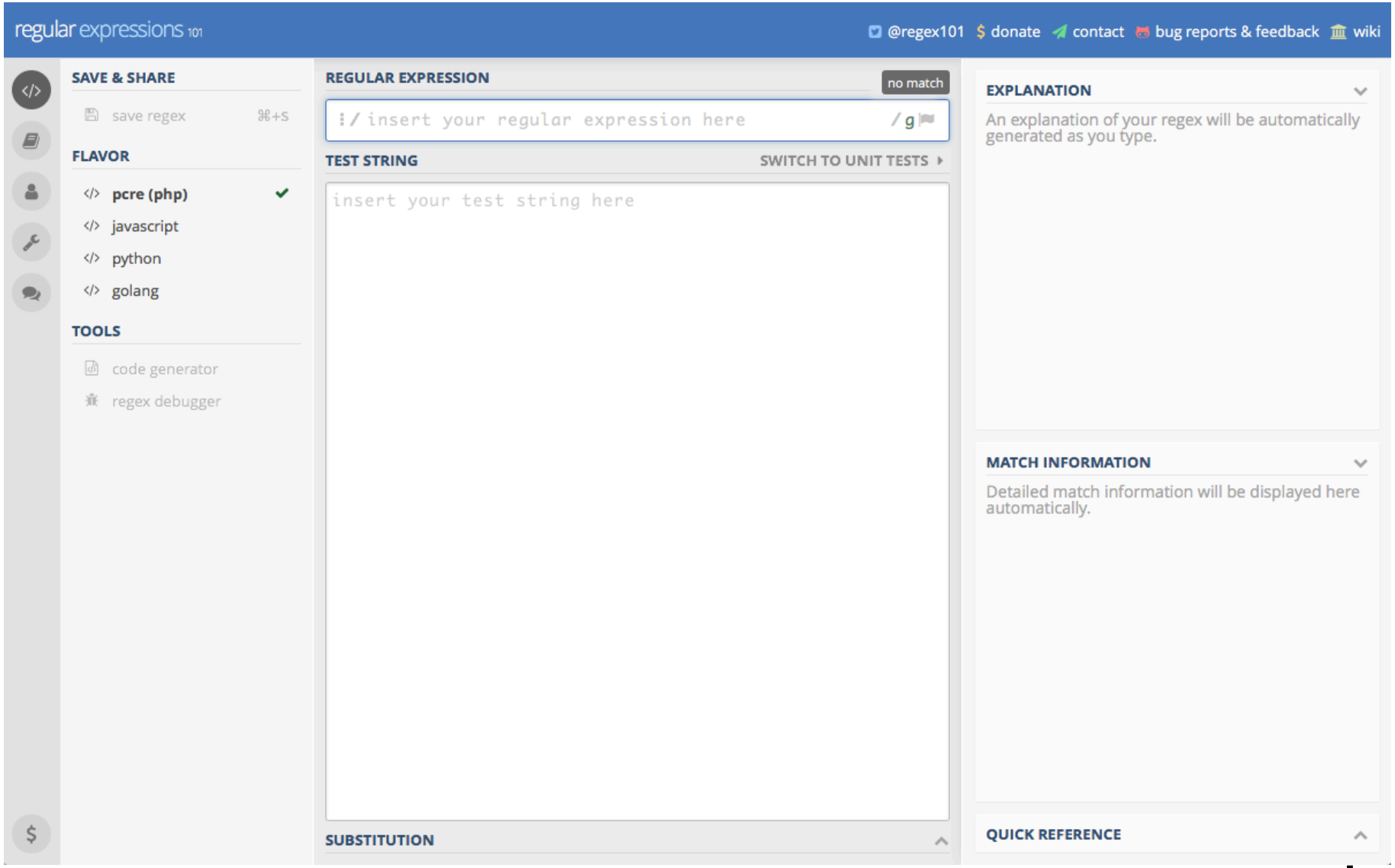
```
Wed Jun 08 2016 09:39:12 www1 sshd[4092]: Failed password for invalid user local from 109.169.
```

Hide Regular Expression

```
^\w+\s+\w+\s+\d+\s+\d+\s+\d+:\d+:\d+\s+\w+\d+\s+\w+\[\d+\]:\s+\w+\s+\w+\s+\w+\s+\w+\s+\w+\s+(?P<user>[^ ]+)
```

splunk> .conf2017

My tool of choice: regex101.com



splunk> .conf2017

Add the data and a regex

REGULAR EXPRESSION

```
/ for (?P<user>\S+) from
```

TEST STRING

```
Mon Jun 06 2016 02:29:19 www1 sshd[3849]: Failed password for root from 128.2
Mon Jun 06 2016 02:29:24 www1 sshd[4267]: Failed password for invalid user ad
Mon Jun 06 2016 02:29:44 www1 sshd[5001]: Failed password for invalid user ge
Mon Jun 06 2016 02:30:13 www1 sshd[1638]: pam_unix(sshd:session): session ope
Mon Jun 06 2016 02:30:13 www1 sshd[1480]: Failed password for invalid user yp
Mon Jun 06 2016 02:30:22 www1 sshd[2291]: Failed password for invalid user em
Mon Jun 06 2016 02:30:26 www1 sshd[4761]: Failed password for invalid user lo
Mon Jun 06 2016 02:30:50 www1 sshd[4986]: Failed password for invalid user my
Mon Jun 06 2016 02:31:19 www1 sshd[81145]: pam_unix(sshd:session): session op
Mon Jun 06 2016 02:31:19 www1 sshd[2021]: Failed password for myuan from 10.1
Mon Jun 06 2016 02:31:28 www1 sshd[1155]: Failed password for invalid user yp
Mon Jun 06 2016 02:29:19 www1 sshd[3849]: Failed password for root from 128.2
Mon Jun 06 2016 02:29:24 www1 sshd[4267]: Failed password for invalid user ad
```

Refine the regex – better matches, but not all

REGULAR EXPRESSION

```
/ for (invalid user )?(?P<user>\S+) from
```

TEST STRING

```
Mon Jun 06 2016 02:29:19 www1 sshd[3849]: Failed password for root from 128.241.220.82 port 2253
Mon Jun 06 2016 02:29:24 www1 sshd[4267]: Failed password for invalid user administrator from 12
Mon Jun 06 2016 02:29:44 www1 sshd[5001]: Failed password for invalid user george from 128.241.2
Mon Jun 06 2016 02:30:13 www1 sshd[1638]: pam_unix(sshd:session): session opened for user djohns
Mon Jun 06 2016 02:30:13 www1 sshd[1480]: Failed password for invalid user yp from 128.241.220.8
Mon Jun 06 2016 02:30:22 www1 sshd[2291]: Failed password for invalid user email from 128.241.22
Mon Jun 06 2016 02:30:26 www1 sshd[4761]: Failed password for invalid user local from 128.241.22
Mon Jun 06 2016 02:30:50 www1 sshd[4986]: Failed password for invalid user mysql from 128.241.22
Mon Jun 06 2016 02:31:19 www1 sshd[81145]: pam_unix(sshd:session): session opened for user djohr
Mon Jun 06 2016 02:31:19 www1 sshd[2021]: Failed password for myuan from 10.1.10.172 port 4468 s
Mon Jun 06 2016 02:31:28 www1 sshd[1155]: Failed password for invalid user yp from 10.1.10.172 p
Mon Jun 06 2016 02:29:19 www1 sshd[3849]: Failed password for root from 128.241.220.82 port 2253
Mon Jun 06 2016 02:29:24 www1 sshd[4267]: Failed password for invalid user administrator from 12
Mon Jun 06 2016 02:29:44 www1 sshd[5001]: Failed password for invalid user george from 128.241.2
Mon Jun 06 2016 02:30:13 www1 sshd[1638]: pam_unix(sshd:session): session opened for user djohns
Mon Jun 06 2016 02:30:13 www1 sshd[1480]: Failed password for invalid user yp from 128.241.220.8
Mon Jun 06 2016 02:30:22 www1 sshd[2291]: Failed password for invalid user email from 128.241.22
Mon Jun 06 2016 02:30:26 www1 sshd[4761]: Failed password for invalid user local from 128.241.22
Mon Jun 06 2016 02:30:50 www1 sshd[4986]: Failed password for invalid user mysql from 128.241.22
Mon Jun 06 2016 02:31:19 www1 sshd[81145]: pam_unix(sshd:session): session opened for user djohr
```

splunk> .conf2017

Refine the regex again – almost there

REGULAR EXPRESSION

```
/ for ((invalid user )|(user ))?(?P<user>\S+) (from|by)
```

TEST STRING

```
Mon Jun 06 2016 02:29:19 www1 sshd[3849]: Failed password for root from 128.241.220.82 port 2253 ssh2
Mon Jun 06 2016 02:29:24 www1 sshd[4267]: Failed password for invalid user administrator from 128.241.22
Mon Jun 06 2016 02:29:44 www1 sshd[5001]: Failed password for invalid user george from 128.241.220.82 po
Mon Jun 06 2016 02:30:13 www1 sshd[1638]: pam_unix(sshd:session): session opened for user djohnson by (u
Mon Jun 06 2016 02:30:13 www1 sshd[1480]: Failed password for invalid user yp from 128.241.220.82 port 2
Mon Jun 06 2016 02:30:22 www1 sshd[2291]: Failed password for invalid user email from 128.241.220.82 por
Mon Jun 06 2016 02:30:24 www1 sudo: djohnson ;  TTY=pts/0 ; PWD=/home/djohnson ; USER=root ; COMMAND=/bi
Mon Jun 06 2016 02:30:26 www1 sshd[4761]: Failed password for invalid user local from 128.241.220.82 por
Mon Jun 06 2016 02:30:50 www1 sshd[4986]: Failed password for invalid user mysql from 128.241.220.82 por
Mon Jun 06 2016 02:31:19 www1 sshd[81145]: pam_unix(sshd:session): session opened for user djohnson by (
Mon Jun 06 2016 02:31:19 www1 sshd[2021]: Failed password for myuan from 10.1.10.172 port 4468 ssh2
Mon Jun 06 2016 02:31:28 www1 sshd[1155]: Failed password for invalid user yp from 10.1.10.172 port 1822
Mon Jun 06 2016 02:29:19 www1 sshd[3849]: Failed password for root from 128.241.220.82 port 2253 ssh2
Mon Jun 06 2016 02:29:24 www1 sshd[4267]: Failed password for invalid user administrator from 128.241.22
Mon Jun 06 2016 02:29:44 www1 sshd[5001]: Failed password for invalid user george from 128.241.220.82 po
```

splunk> .conf2017

And FINALLY – we got them all

Four different formats – all four user field types found!

REGULAR EXPRESSION

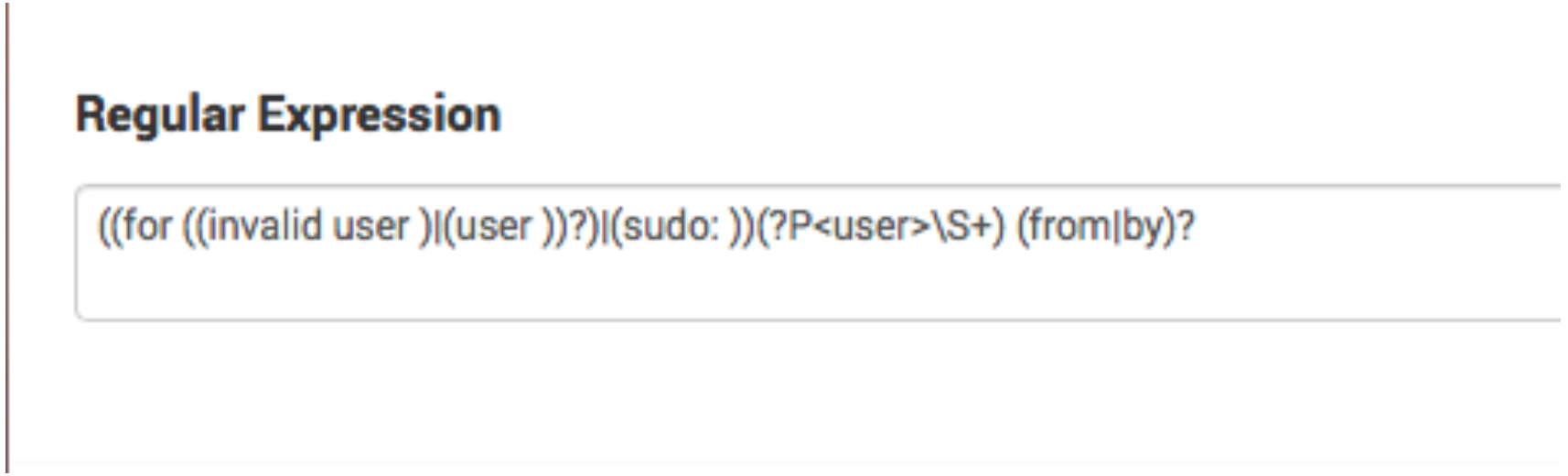
```
/ ((for ((invalid user )|(user ))?)|(sudo: ))(?P<user>\S+) (from|by)?
```

TEST STRING

```
Mon Jun 06 2016 02:29:19 www1 sshd[3849]: Failed password for root from 128.241.220.82 port 2253 ssh2
Mon Jun 06 2016 02:29:24 www1 sshd[4267]: Failed password for invalid user administrator from 128.241.22
Mon Jun 06 2016 02:29:44 www1 sshd[5001]: Failed password for invalid user george from 128.241.220.82 po
Mon Jun 06 2016 02:30:13 www1 sshd[1638]: pam_unix(sshd:session): session opened for user djohnson by (u
Mon Jun 06 2016 02:30:13 www1 sshd[1480]: Failed password for invalid user yp from 128.241.220.82 port 2
Mon Jun 06 2016 02:30:22 www1 sshd[2291]: Failed password for invalid user email from 128.241.220.82 por
Mon Jun 06 2016 02:30:24 www1 sudo: djohnson ;  TTY=pts/0 ; PWD=/home/djohnson ; USER=root ; COMMAND=/bi
Mon Jun 06 2016 02:30:26 www1 sshd[4761]: Failed password for invalid user local from 128.241.220.82 por
Mon Jun 06 2016 02:30:50 www1 sshd[4986]: Failed password for invalid user mysql from 128.241.220.82 por
Mon Jun 06 2016 02:31:19 www1 sshd[81145]: pam_unix(sshd:session): session opened for user djohnson by (
Mon Jun 06 2016 02:31:19 www1 sshd[2021]: Failed password for myuan from 10.1.10.172 port 4468 ssh2
Mon Jun 06 2016 02:31:28 www1 sshd[1155]: Failed password for invalid user yp from 10.1.10.172 port 1822
Mon Jun 06 2016 02:29:19 www1 sshd[3849]: Failed password for root from 128.241.220.82 port 2253 ssh2
```

Back to the FET – use our regex

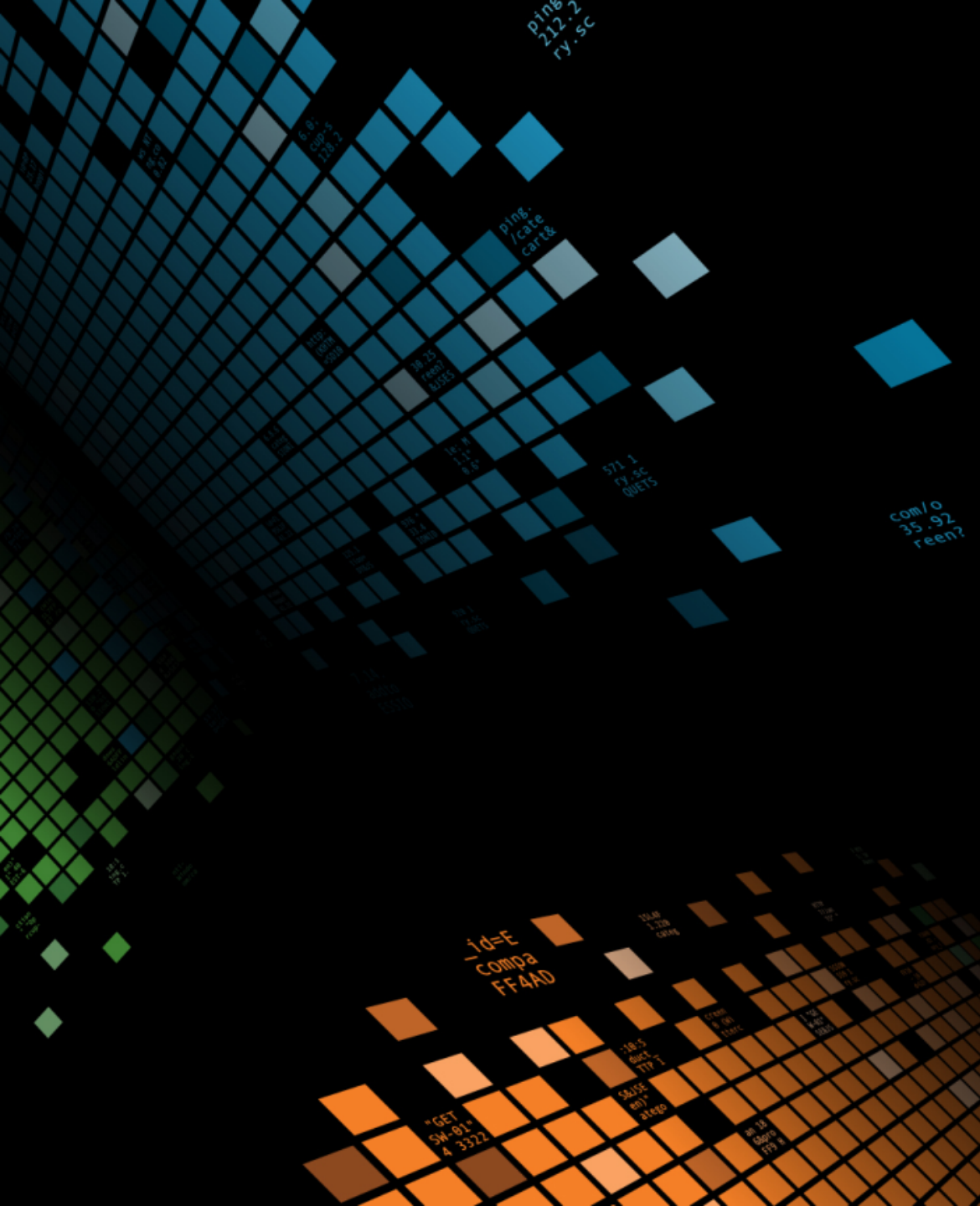
Paste this regex into the field that the ugly, single-format regex was in:



…and the results are MUCH better

```
www1 sshd[3985]: Failed password for invalid user testuser from 10.1.10.172 por
www1 sshd[3697]: Failed password for invalid user whois from 10.1.10.172 port 1
www1 sudo: djohnson ;  TTY=pts/0 ; PWD=/home/djohnson ; USER=root ; COMMAND=/bi
www1 sshd[4333]: Failed password for root from 10.1.10.172 port 2772 ssh2
www1 sshd[1632]: Failed password for invalid user elena_andubasquet from 10.1.1
www1 sshd[1155]: Failed password for invalid user yp from 10.1.10.172 port 1822
www1 sshd[2021]: Failed password for myuan from 10.1.10.172 port 4468 ssh2
www1 sshd[81145]: pam_unix(sshd:session): session opened for user djohnson by (
www1 sshd[4986]: Failed password for invalid user mysql from 128.241.220.82 por
www1 sshd[4761]: Failed password for invalid user local from 128.241.220.82 por
www1 sshd[2291]: Failed password for invalid user email from 128.241.220.82 por
www1 sshd[1480]: Failed password for invalid user yp from 128.241.220.82 port 2
www1 sshd[1638]: pam_unix(sshd:session): session opened for user djohnson by (u
www1 sshd[5001]: Failed password for invalid user george from 128.241.220.82 po
```

splunk> .conf2017



How’d He Do That?

Tricks for getting it right

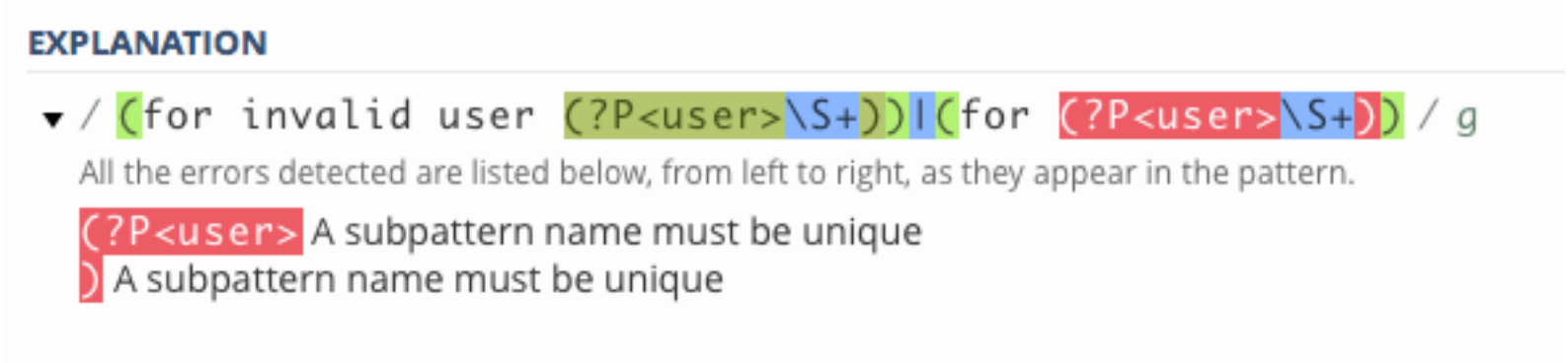
splunk> .conf2017

One named capture group with a single name

More than one instance of the same name will fail

- (for invalid user (?P<user>\S+))|(for (?P<user>\S+))

- Capture group names must be unique:



How do you eat an elephant?

Bite by bite is better than trying to stuff the whole elephant in your mouth at once

- Start with one format
- Try to find similarities and differences between the formats
- Add a new format to your data and check your updated regex
- Keep a copy of the last one that worked!!
- Add additional formats and check ALL matches for ALL examples

splunk> .conf2017

Alternate: Do one for each format

Then you can try combining

- This can be more difficult with simpler regexes
- Can be easier for more complex regexes
- Combine two of the regexes that are similar
- Try to keep the things that are the same in both, making the changes to the original only where there is a difference
- Remember – **ONE instance of a name per regex**

Use Parentheses!

Make it easier to come back to later

((for ((invalid user)|(user))?)|(sudo:))(?P<user>\S+) (from|by)?

- Using parentheses for clarity is helpful:
- They make it possible to see the separate parts and their relationships with each other
- Don’t overdo the parentheses

Use the Best Character Class

Use the right tool for the job

- Sometimes a field regex must be able to match data that hasn't been seen in the data yet, so in this case **be as general as possible**

- In the previous example, the **\S** is best because **\s** will be the delimiter (a space in this case) because you want to catch any potential case that you don't see in the data, *yet*.

\S+

- If you have a delimiter that you can count on, use something like this to match the field value (**in this case be specific** about what it is NOT):

[^,]+

Just because your sample data doesn't have a particular character in it, that doesn't mean it never will. Examples:

- Usernames – alphanumerics + what?
  - Dash – Underscore - Other characters - Are you sure?
- Filenames – could you have a space?
  - C:\Program Files\My Application\

splunk> .conf2017

If you have a definite delimiter, take advantage

Examples:

- Data: ”contents of quoted string”
  - Use: ”(?P<contents>[^”]+)”
- Data: User:carypetterborg Dept:ICS
  - Use: User:(?P<user>\S+)\s
- Data: Salt Lake City, Utah 84117-6403
  - Use: ^(?P<city>[^,]+),\s+(?P<state>.+)\s+(?P<zip>[-\d]*)$

splunk> .conf2017

REX and REGEX Commands

The most common use for regular expressions is in SPL with ***rex*** and ***regex***

splunk> .conf2017

When to Use REX

When you:

- **don’t always** want to extract the data
- want to extract data from a field that is already extracted
- don’t have access to field extractions (permissions, etc.)
- require doing multiple, disparate regular expressions
- are in a hurry or you are doing a proof-of-concept

REX Example

index=voice sourcetype=voice* Description | rex "Description=(?P<description>[^\^]+)" | rex field=description "From (?P<start>.+) to (?P<end>.+?):\s”

Aug 2 20:32:28 I13772 CPCMI13772: %local7-2-ALARM: 16$Description= Number of AuthenticationFailed events exceeds configured threshold during configured interval of time 1 within 3 minutes on cluster StandAloneCluster. There are 2 AuthenticationFailed events (up to 30) received during the monitoring interval From **Wed Aug 03 10:25:00 PHT 2016** to **Wed Aug 03 10:28:00 PHT 2016**: TimeStamp : 8/3/16 10:26 AM LoginFrom : 172.12.34.40 Interface : VMREST UserID : JacobMD AppID : Cisco Tomcat ClusterID : NodeID : APPHMANAOVM001 TimeStamp : Wed Aug 03 10:26:13 PHT 2016 TimeStam::Status=2,cleared^Severity=minor^Acknowledged=no^CUSTOMER=Cisco Prime Collaboration^Private IP Address=172.16.17.24^Default Alarm Name=AuthenticationFailed^Managed Object=10.160.17.24^Managed Object Type=Unity Connection^MODE=2;Alarm ID=343815480^Component=10.160.17.24\x00000

splunk> .conf2017

REX Commands

It takes two:

- index=voice sourcetype=voice* Description
  | rex "Description=(?P<description>[^\^]+)"
  | rex field=description "From (?P<start>.+) to (?P<end>.+?):\s”

- SYNTAX:
  | rex [field=*fieldname*] “regex”

- Also available:
  | rex mode=sed

splunk> .conf2017

First rex – get the description

REGULAR EXPRESSION

1 match, 23 steps (~5ms)

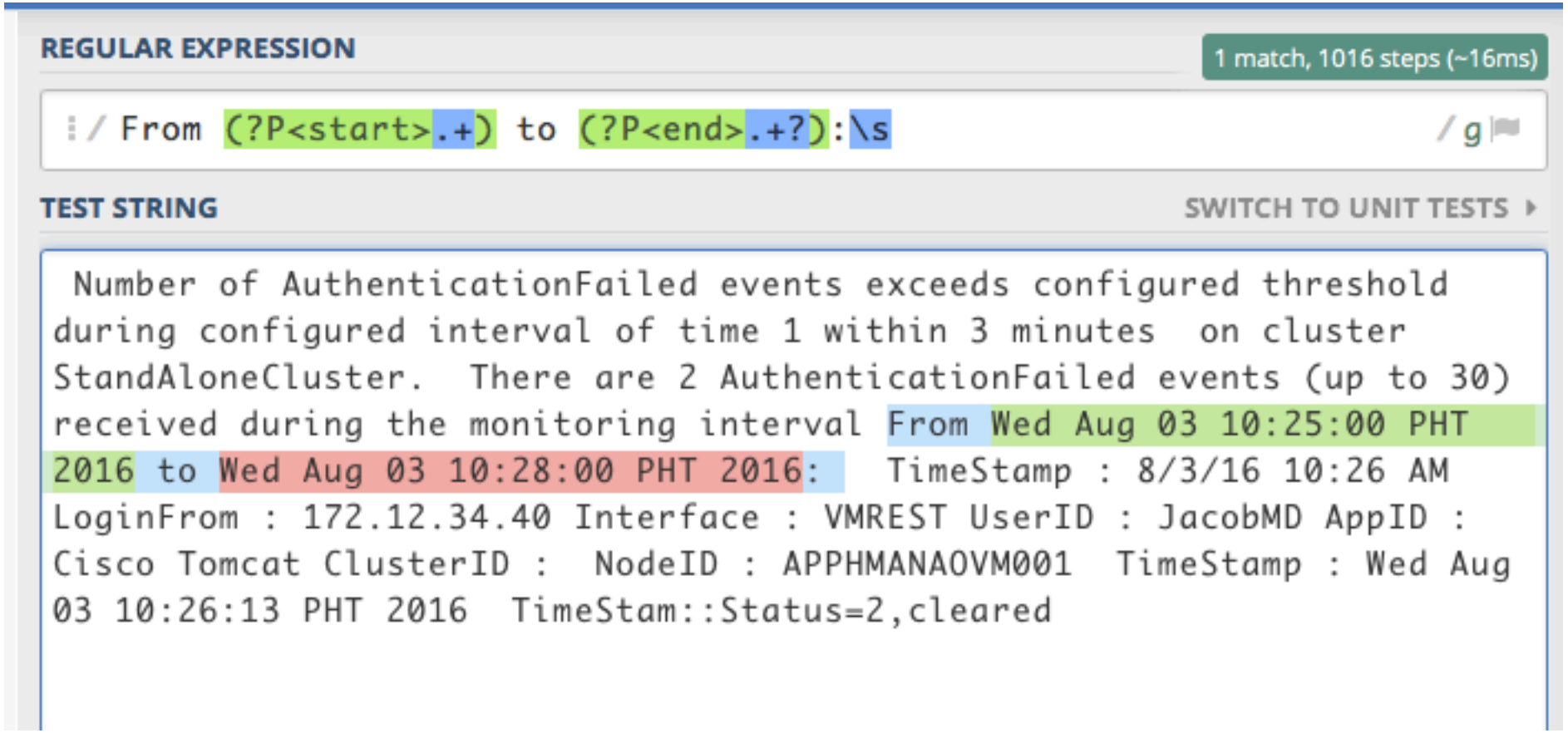
```
/ Description=(?P<description>[^\^]+)                                    /g
```

TEST STRING

SWITCH TO UNIT TESTS

```
Aug  2 20:32:28 l13772 CPCMl13772: %local7-2-ALARM: 16$Description= Number of
AuthenticationFailed events exceeds configured threshold during configured interval of
time 1 within 3 minutes  on cluster StandAloneCluster.  There are 2
AuthenticationFailed events (up to 30) received during the monitoring interval From Wed
Aug 03 10:25:00 PHT 2016 to Wed Aug 03 10:28:00 PHT 2016:   TimeStamp : 8/3/16 10:26 AM
LoginFrom : 172.12.34.40 Interface : VMREST UserID : JacobMD AppID : Cisco Tomcat
ClusterID :  NodeID : APPHMANAOVM001  TimeStamp : Wed Aug 03 10:26:13 PHT 2016
TimeStam::Status=2,cleared^Severity=minor^Acknowledged=no^CUSTOMER=Cisco Prime
Collaboration^Private IP Address=172.16.17.24^Default Alarm
Name=AuthenticationFailed^Managed Object=10.160.17.24^Managed Object Type=Unity
Connection^MODE=2;Alarm ID=343815480^Component=10.160.17.24\x00000
```

Second rex – get the Start and End

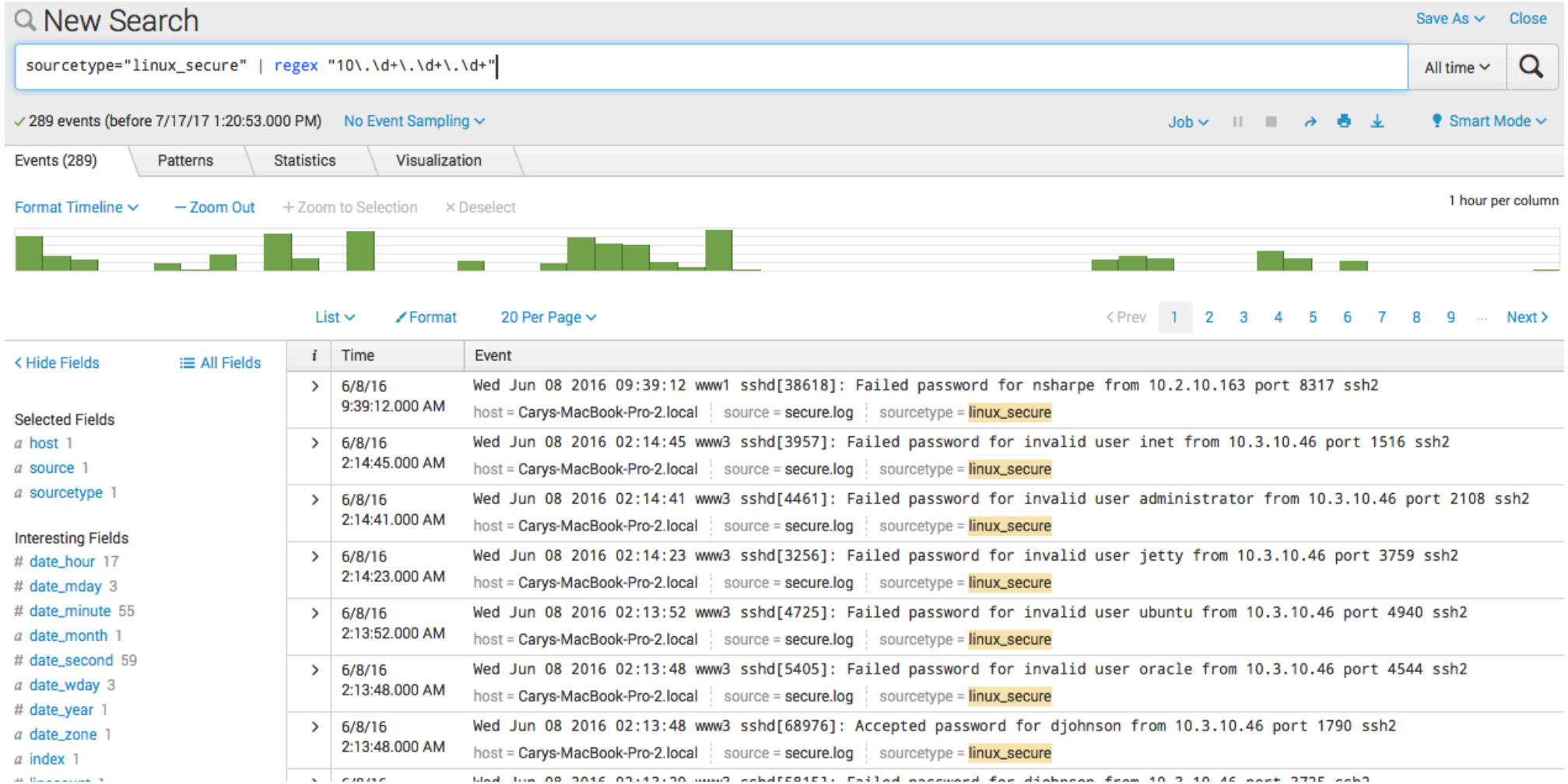


When to use REGEX

- To filter out events/data that you don't want included in the pipeline
- This is like **search** on steroids, but doesn't replace **search**
- Only used as a filter

Regex example

Only get events with internal addresses



splunk> .conf2017

Breakdown

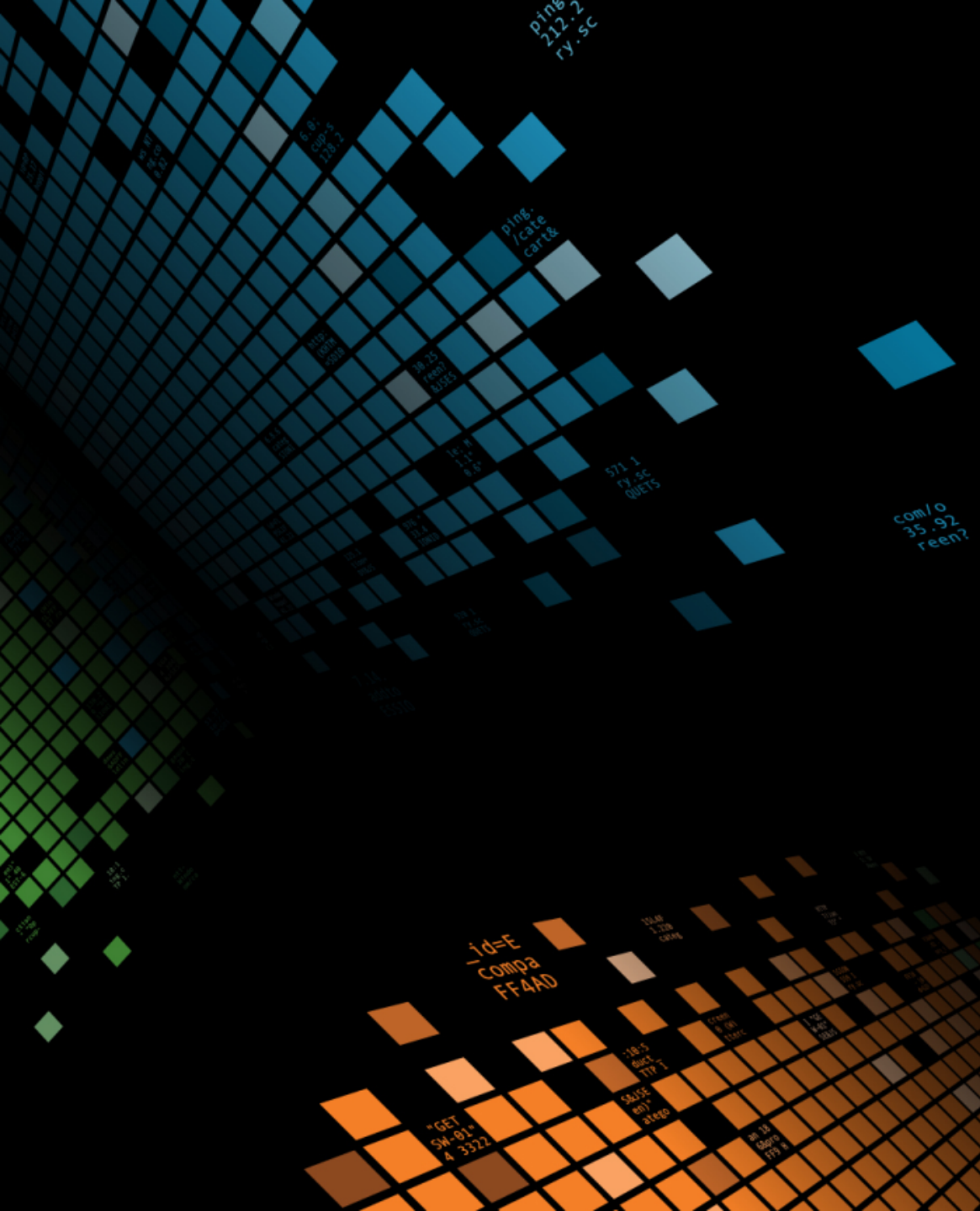
- Search:
  sourcetype=linux_secure | regex "10\.\d+\.\d+\.\d+”
- Only internal (10.*) IP addresses make it through the **regex** filter
- Search produces events, regex then limits those results passed on through the pipeline by a fancy regular expression
- *Yes, there are other ways to do this, but this is a regex example*

Rex vs Regex

- Use **rex** to extract fields
- Use **regex** to limit results
- Yes, you can use them in the same search:

```
sourcetype=linux_secure | rex "from (?P<src_ip>\d+\.\d+\.\d+\.\d+)" | regex src_ip="(?<!10)\.\d+\.\d+\.\d+"
```

splunk> .conf2017



Index Time Regular Expression Usage

splunk> .conf2017

The Problem

- You can’t index Social Security Numbers
- How do you distinguish a Social Security Number from other numbers?
- Obfuscate ONLY SSNs, but leave other things alone.

SSN vs Phone

Regex distinctions

SSN

- 123-45-6789
- \d{3}-\d{**2**}-\d{4}

Phone

- 800-123-4567
- \d{3}-\d{**3**}-\d{4}

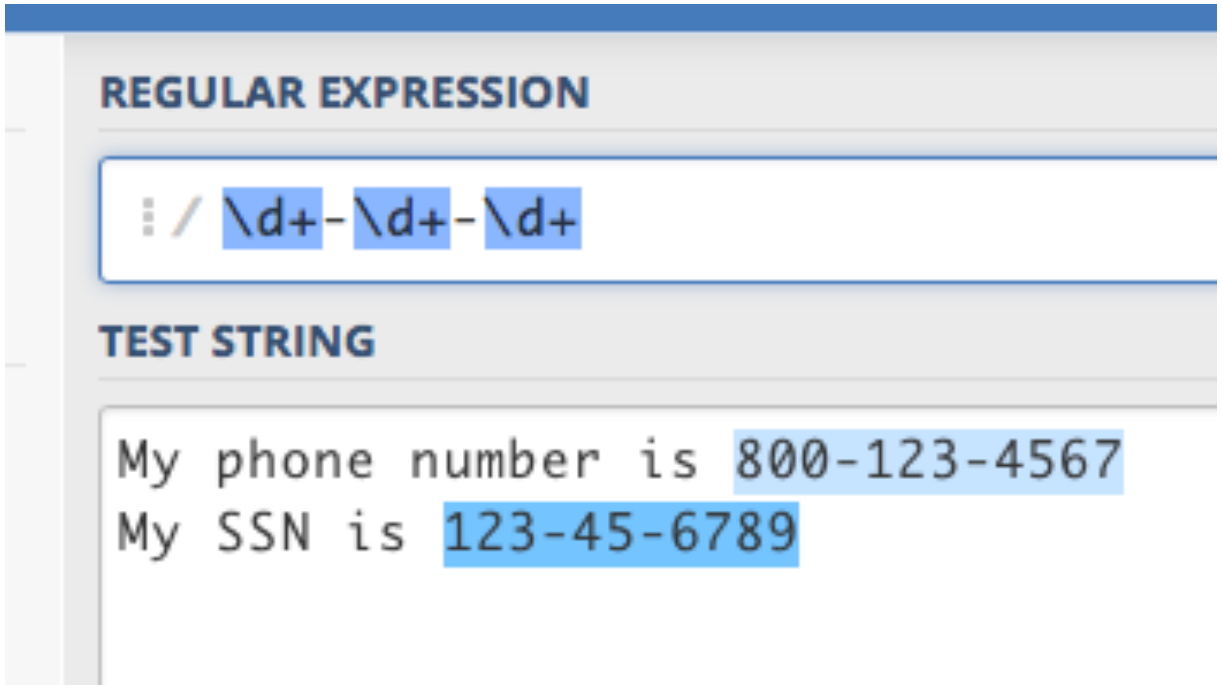
splunk> .conf2017

Be as specific in your matches as possible

- You could use something simple like:

\d+-\d+-\d+

- But it will mistake a phone number for a SSN:

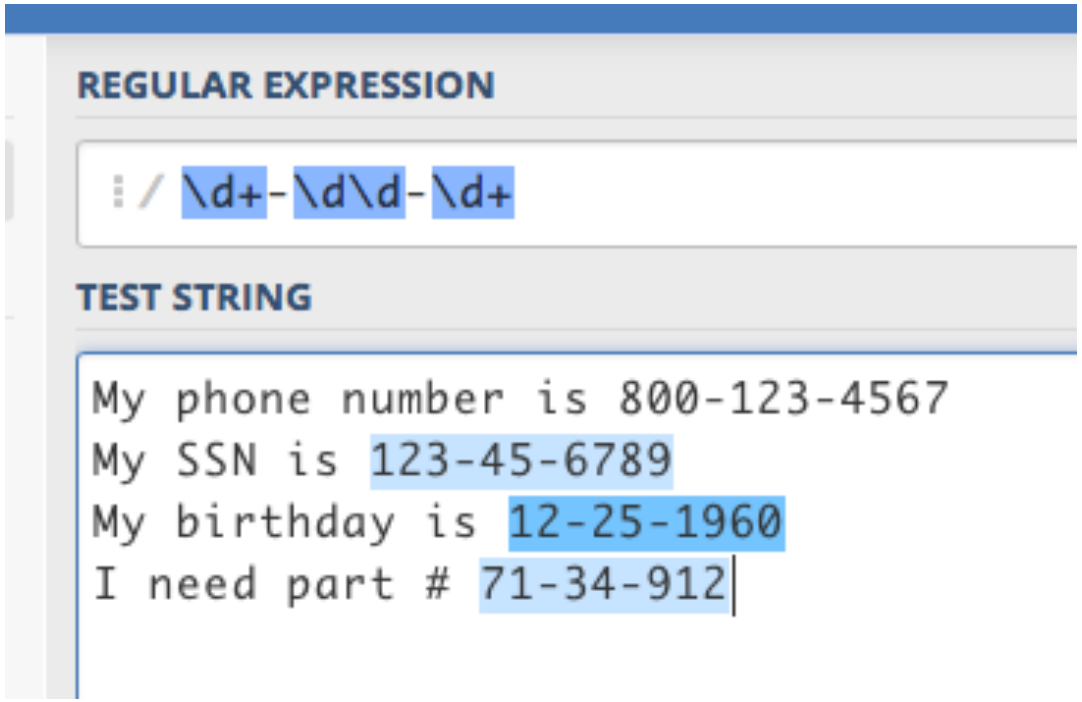


A Better Match

- New regex:

\d+-\d\d-\d+

- Gets rid of Phone #'s, but what about other data?

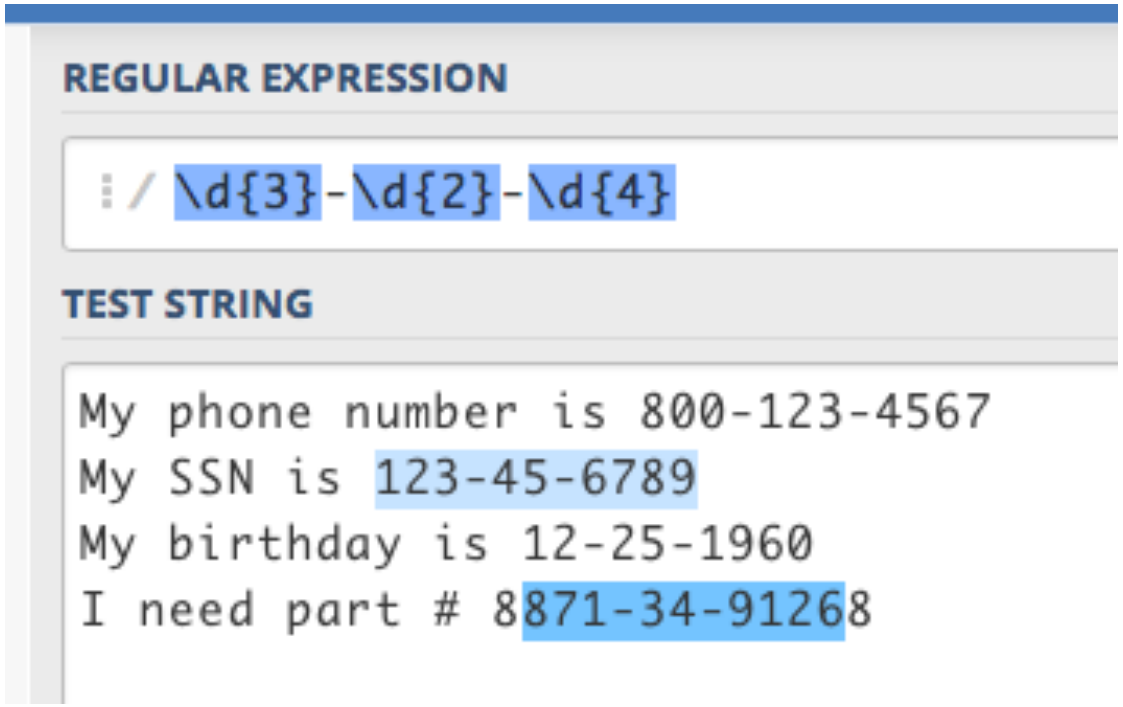


We’re now so close

- This is exactly what a properly formatted SSN looks like:

\d{3}-\d{2}-\d{4}

- This defines a SSN, but it matches other things, too:

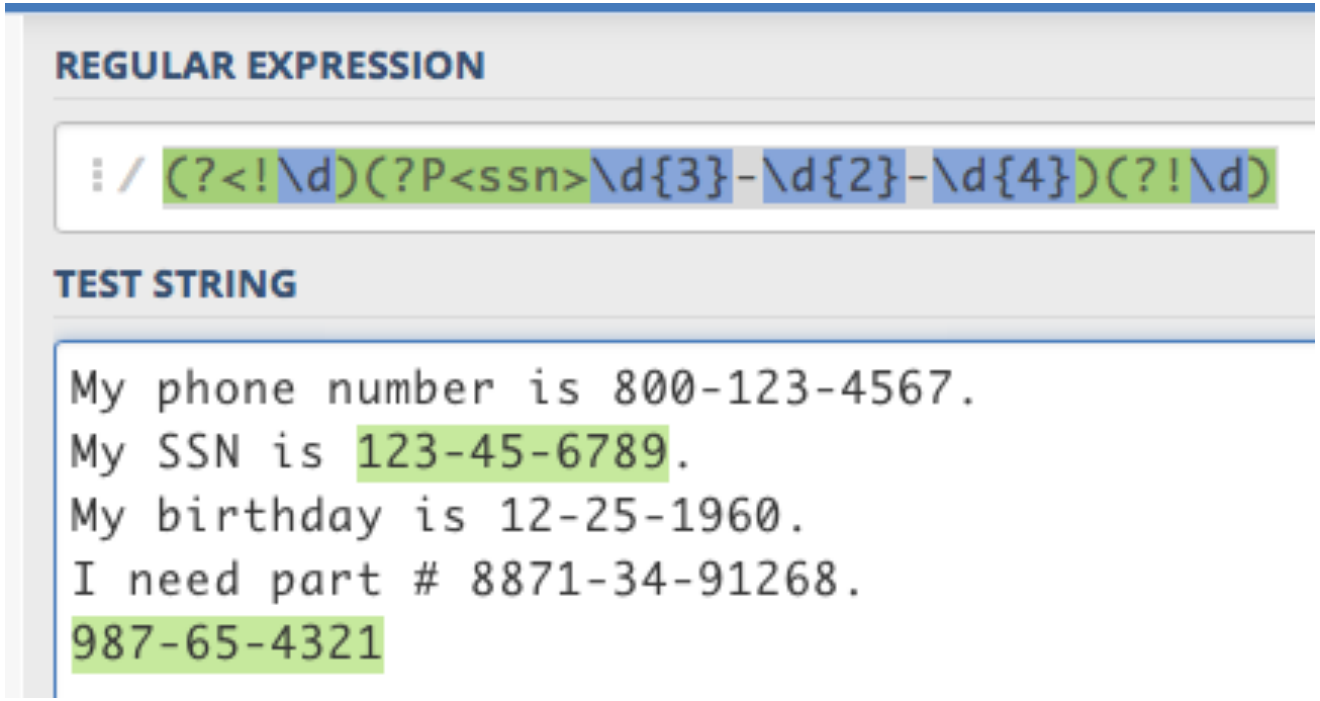


Let's get REAL specific

- Best definition:

(?<!\d)(?P<ssn>\d{3}-\d{2}-\d{4})(?!\d)

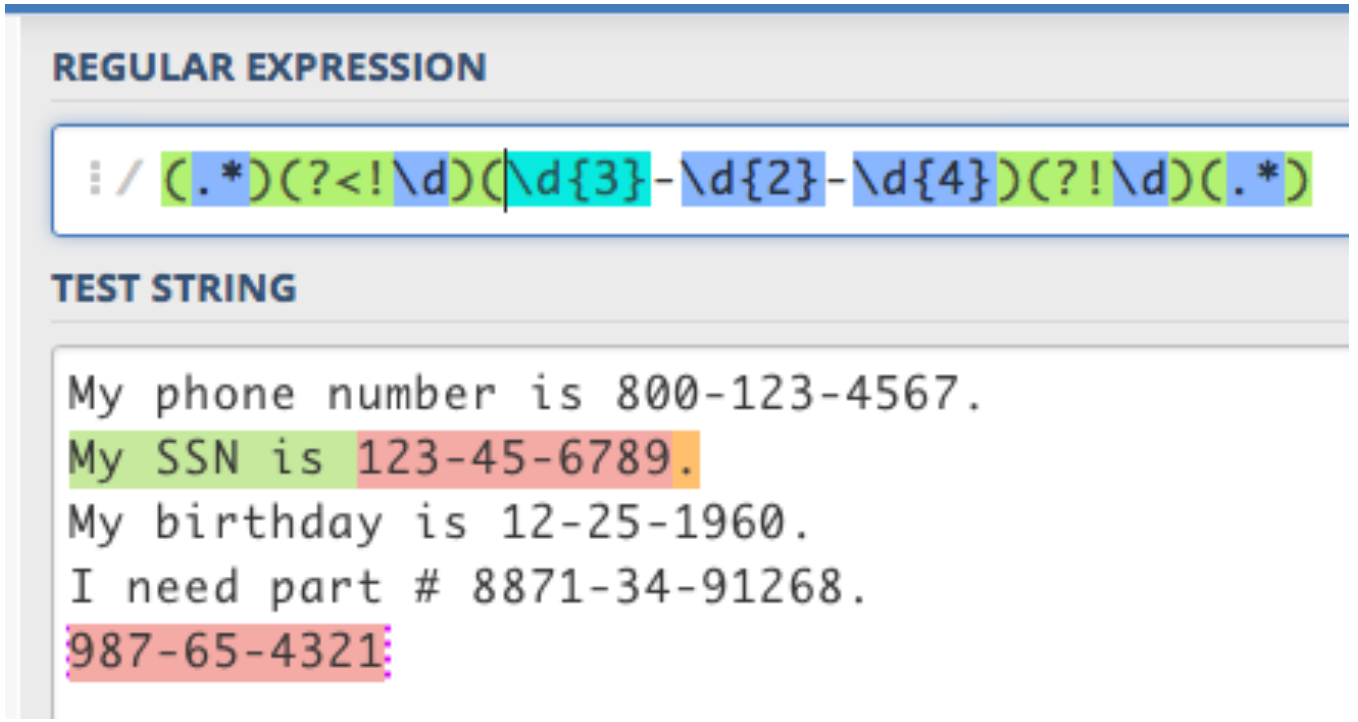
- The SSN match can be found *anywhere* in the event, and only the SSN:



So let's make it work in transforms.conf

- Grab the beginning and ending text of the event:

(.*)(?<!\d)(\d{3}-\d{2}-\d{4})(?!\d)(.*)



Index-time conversion

Conditions and Limitations

- This regex can’t be done with SEDCMD in props.conf alone
  - The regex uses regex features not found in SED format
- Using a simple custom sourcetype, but it can be made a general transform
- Must capture all parts of the event.
- Will obfuscate only one SSN per event.

The props.conf

```
[testssn]
TRANSFORM-ssn = nossn
DATETIME_CONFIG = CURRENT
NO_BINARY_CHECK = true
SHOULD_LINEMERGE = false
disabled = false
```

splunk> .conf2017

The transforms.conf

```
[nossn]
REGEX=(?m)(.*)(?<!\d)(\d{3}-\d{2}-\d{4})(?!\d)(.*)
FORMAT = $1###-##-####$3
DEST_KEY = _raw
```

splunk> .conf2017

New regex features shown

- **(?m)** – perform the regex on multi-line events
- **(?<!\d)** - Negative Lookbehind – *not preceeded by a digit* – no net character
- **(?!\d)** – Negative Lookahead – *not followed by a digit* – no net character

How the FORMAT works

- **FORMAT = $1###-##-###$3**
- **$1** and **$3** are capture group matches – from **(.*)** at beginning and end
- **$2** is not used in the FORMAT, but it's the capture group for the SSN – from:

  (\d{3}-\d{2}-\d{4})

splunk> .conf2017

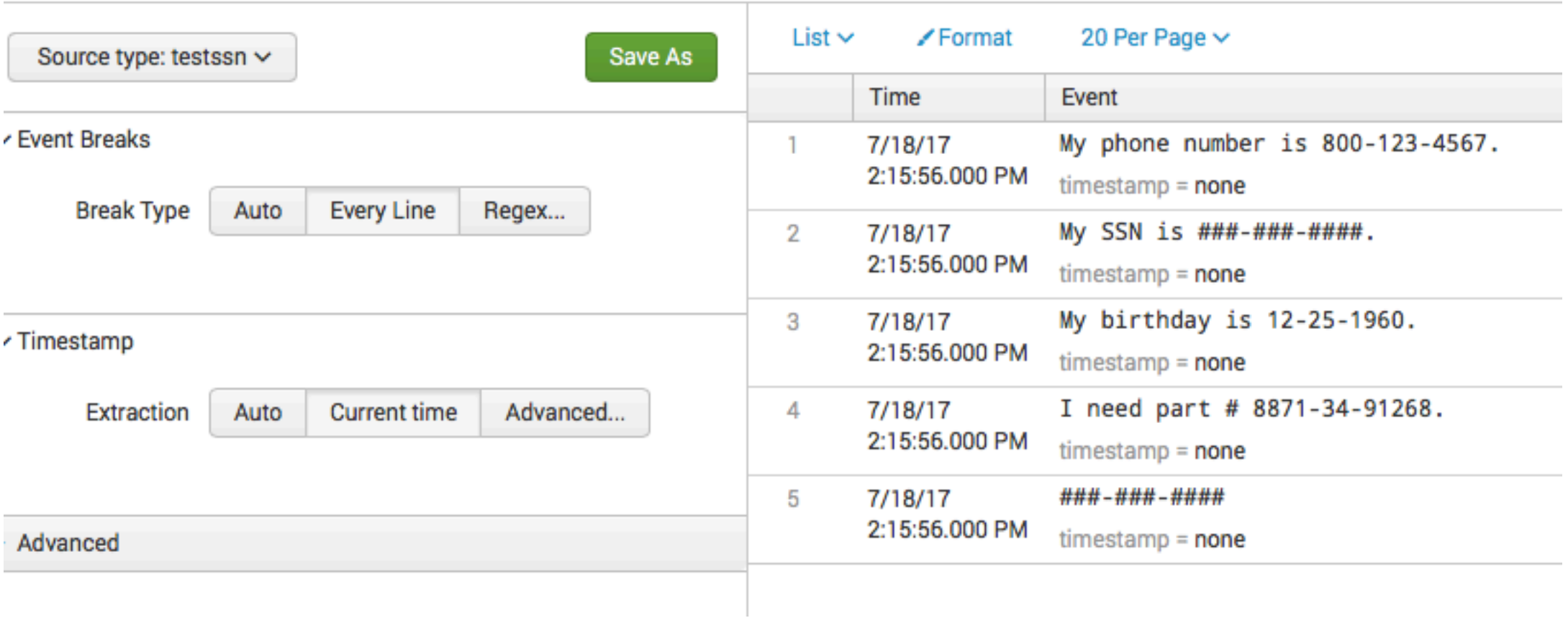
After bringing in the data:

List ▾ Format 20 Per Page ▾

i	Time	Event
>	7/18/17 2:16:51.000 PM	###-###-#### host = Carys-MacBook-Pro-2.local source = ssn.log sourcetype = testssn
>	7/18/17 2:16:51.000 PM	I need part # 8871-34-91268. host = Carys-MacBook-Pro-2.local source = ssn.log sourcetype = testssn
>	7/18/17 2:16:51.000 PM	My birthday is 12-25-1960. host = Carys-MacBook-Pro-2.local source = ssn.log sourcetype = testssn
>	7/18/17 2:16:51.000 PM	My SSN is ###-###-####. host = Carys-MacBook-Pro-2.local source = ssn.log sourcetype = testssn
>	7/18/17 2:16:51.000 PM	My phone number is 800-123-4567. host = Carys-MacBook-Pro-2.local source = ssn.log sourcetype = testssn

splunk> .conf2017

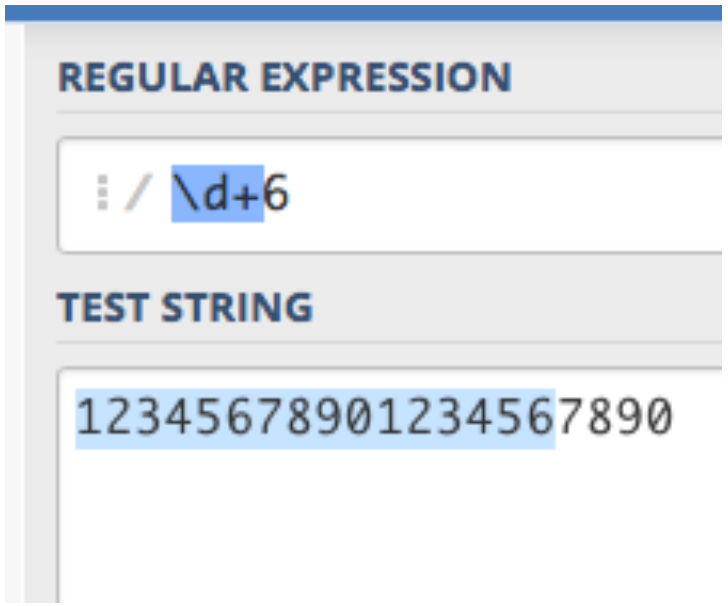
It even will obfuscate the data import preview:

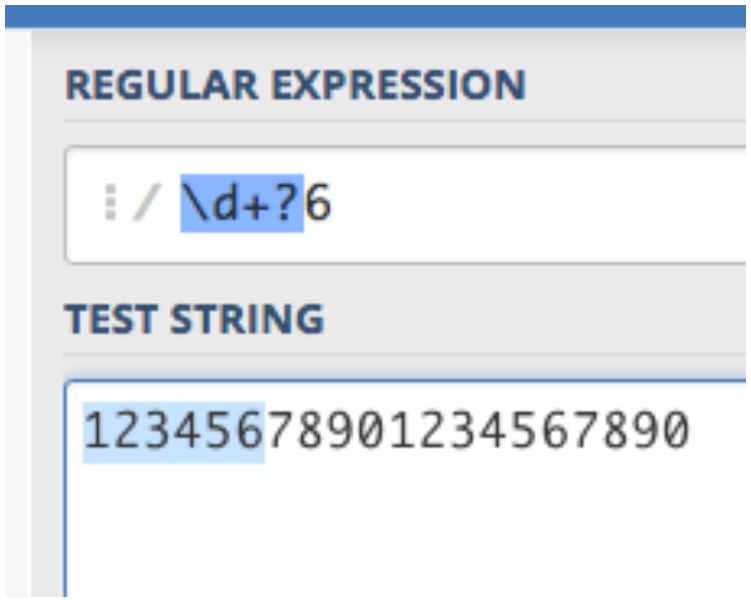


Greedy vs. Lazy Matches

splunk> .conf2017

Greedy vs Lazy





What is the difference?

Subtle difference, but big effect

- **Greedy** – Grab as much as you can
- **Lazy** – Grab as little as you can

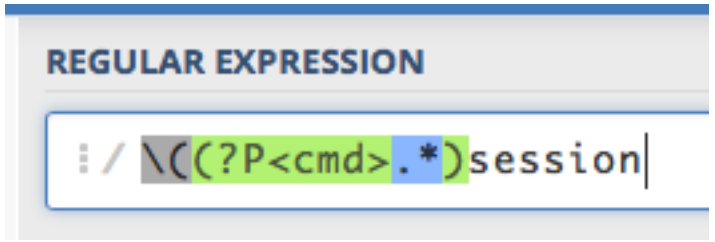
- The lazy match will continue only as far as it needs to, no further
  - **<.+?>** will match **<12345>**, while
  - **<.+>** will match both **<12345>** and **<12345><67890>**

SYNTAX: place a **?** After a ***** or **+**

The lazy match only goes to the first instance of a match following a multiple match

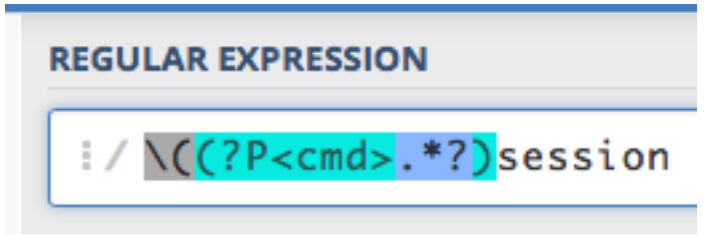
splunk> .conf2017

Greedy Example



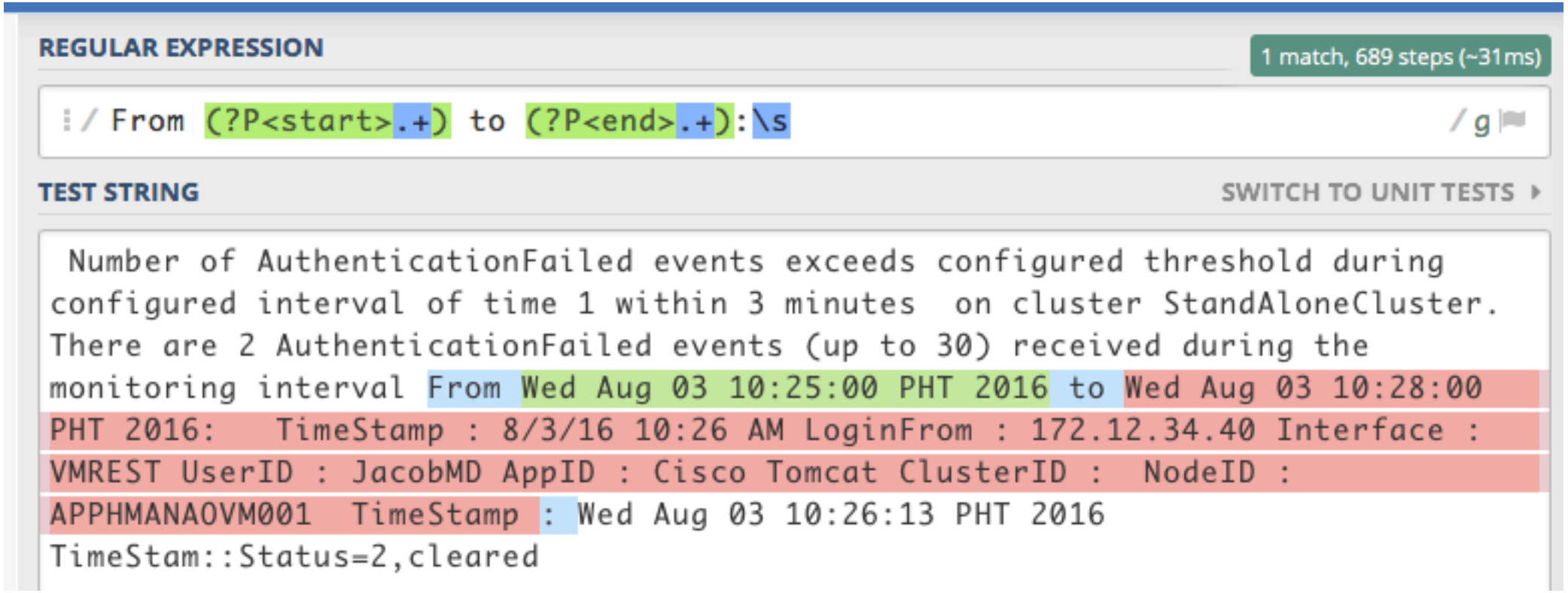

```
led password for invalid user geo
_unix(sshd:session): session open
led password for invalid user yp
led password for invalid user ema
led password for invalid user loc
led password for invalid user mys
m_unix(sshd:session): session ope
led password for myuan from 10.1.
```

Lazy Example

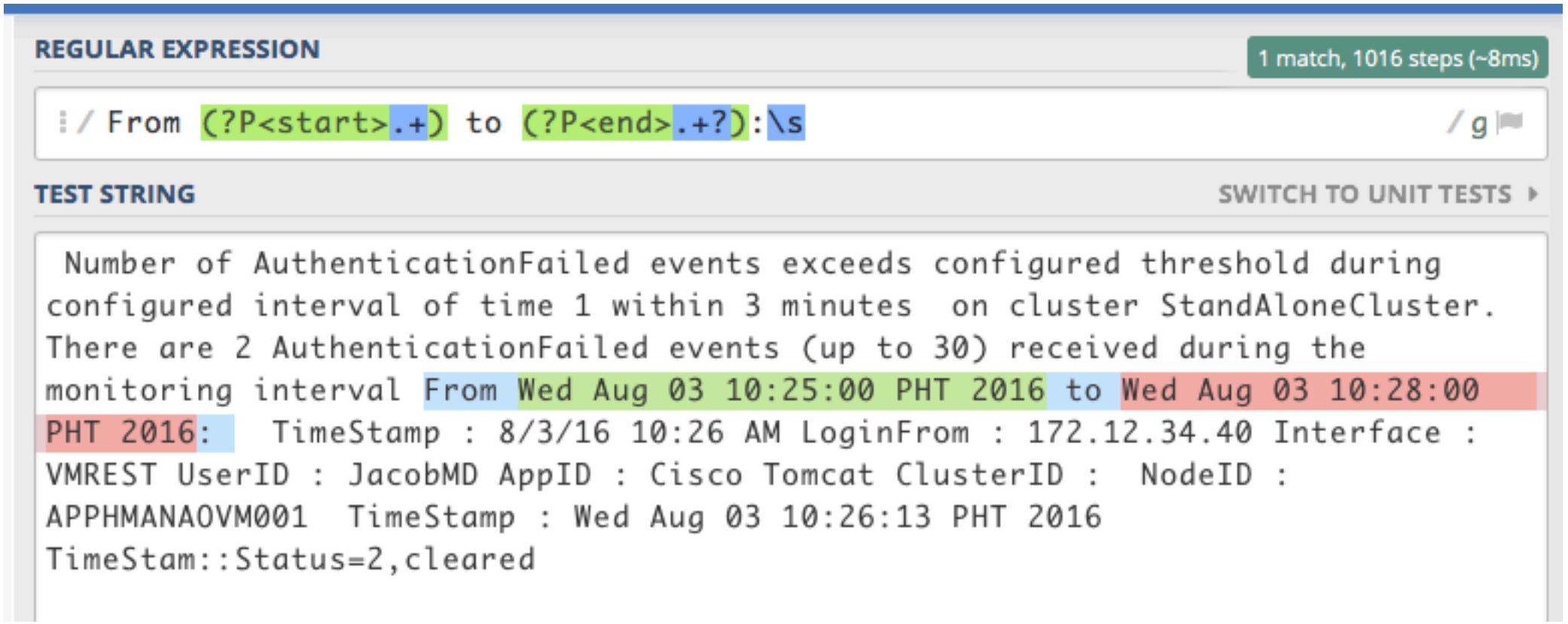



```
ed password for invalid user ge
unix(sshd:session): session ope
ed password for invalid user yp
ed password for invalid user em
ed password for invalid user lo
ed password for invalid user my
_unix(sshd:session): session op
ed password for myuan from 10.1
```

Second Look - Greedy

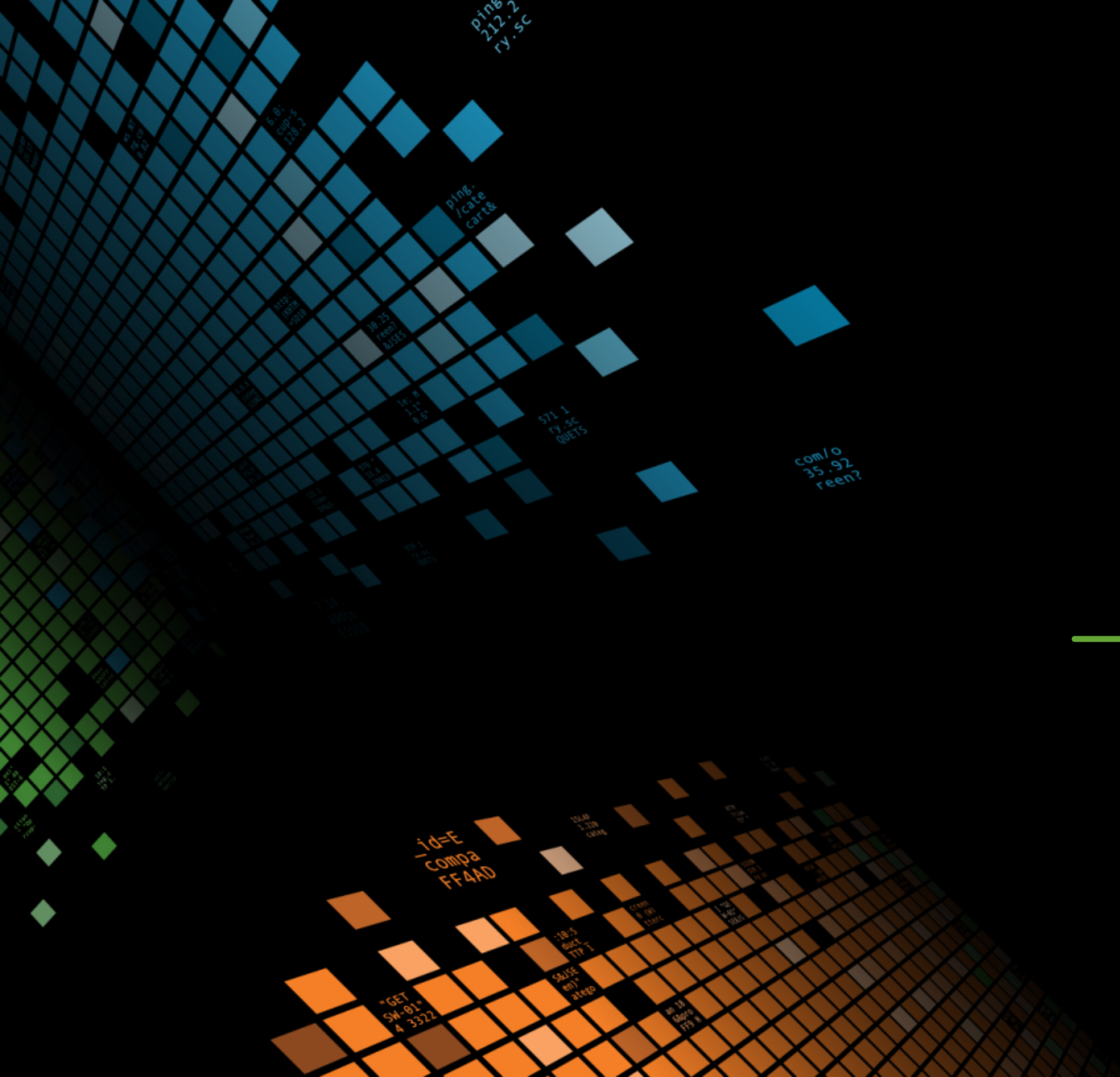


Second Look - Lazy



Choose Wisely

- Greedy may cross long segments
- Lazy may stop prematurely
- **Try it on various data sets** to make sure it will do what you want



Embedding

Multiple field extractions from one piece of data

- Problem:
  - Extract two different fields from the exact same piece of data
  - Only want to use one regex – for efficiency if nothing else
  - Need both the Domain only and the whole URL from an access log

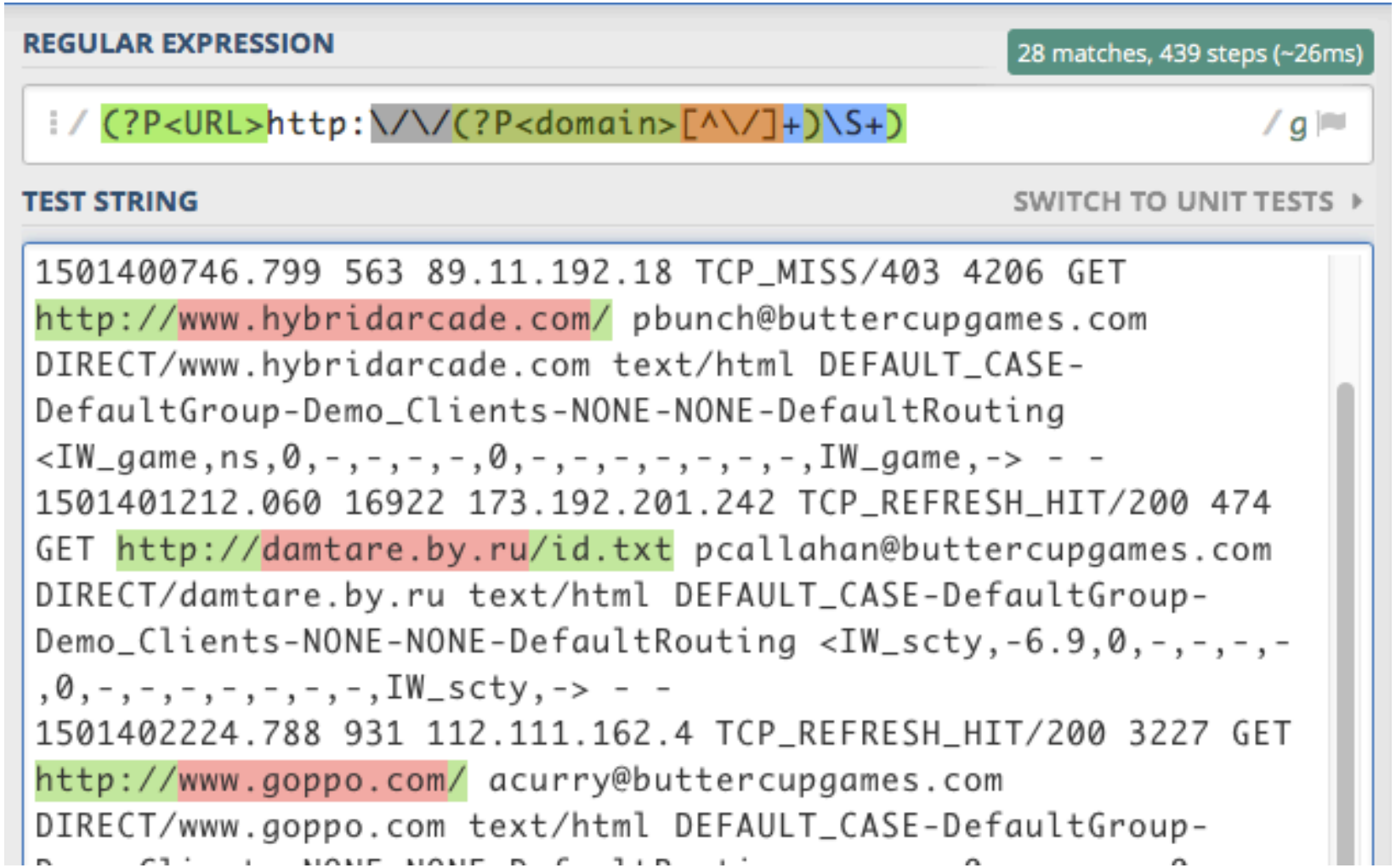
Source and Results

- Data:
  - 1501408932.060 16922 108.65.113.83 TCP_REFRESH_HIT/200 474 GET http://damtare.by.ru/id.txt myuan@buttercupgames.com DIRECT/damtare.by.ru text/html DEFAULT_CASE-DefaultGroup-Demo_Clients-NONE-NONE-DefaultRouting <IW_scty,-6.9,0,-,-,-,-,0,-,-,-,-,-,-,-,IW_scty,-> - -

- Desired Fields:
  - Domain: damtare.by.ru
  - URL: http://damtare.by.ru/id.txt

splunk> .conf2017

Regex



splunk> .conf2017

Regex101.com vs Splunk

- Slashes need escaping in regex101, but not in Splunk:

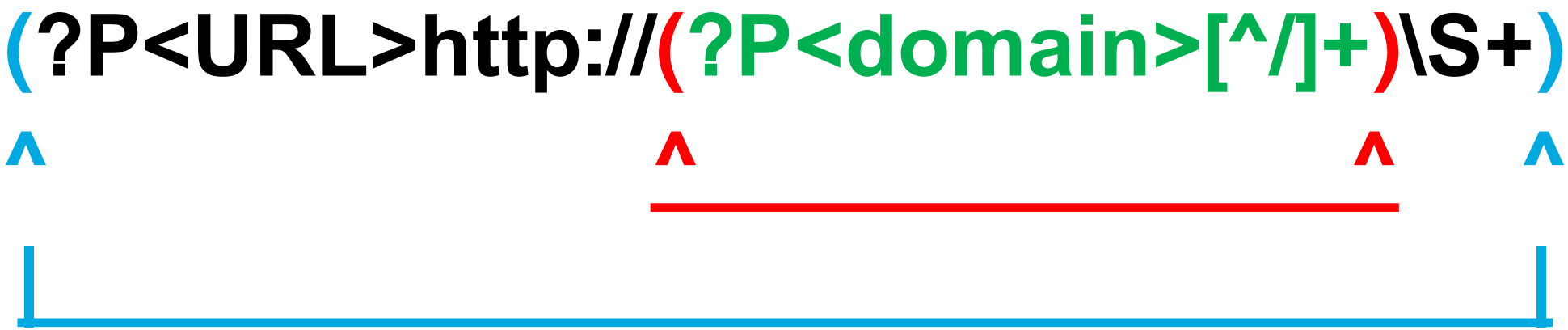
(?P<URL>http:\/\/(?P<domain>[^\/]+)\S+)

vs

(?P<URL>http://(?P<domain>[^/]+)\S+)

splunk> .conf2017

What? … Oh, now I see.



Results in Splunk

You can’t do this in the FET without doing your own regex!

Regular Expression

Regular Expression Reference ↗ View in Search ↗

```
(?P<URL>http://(?P<domain>[^/]+)\S+)
```

Preview | Save

Events | URL | domain

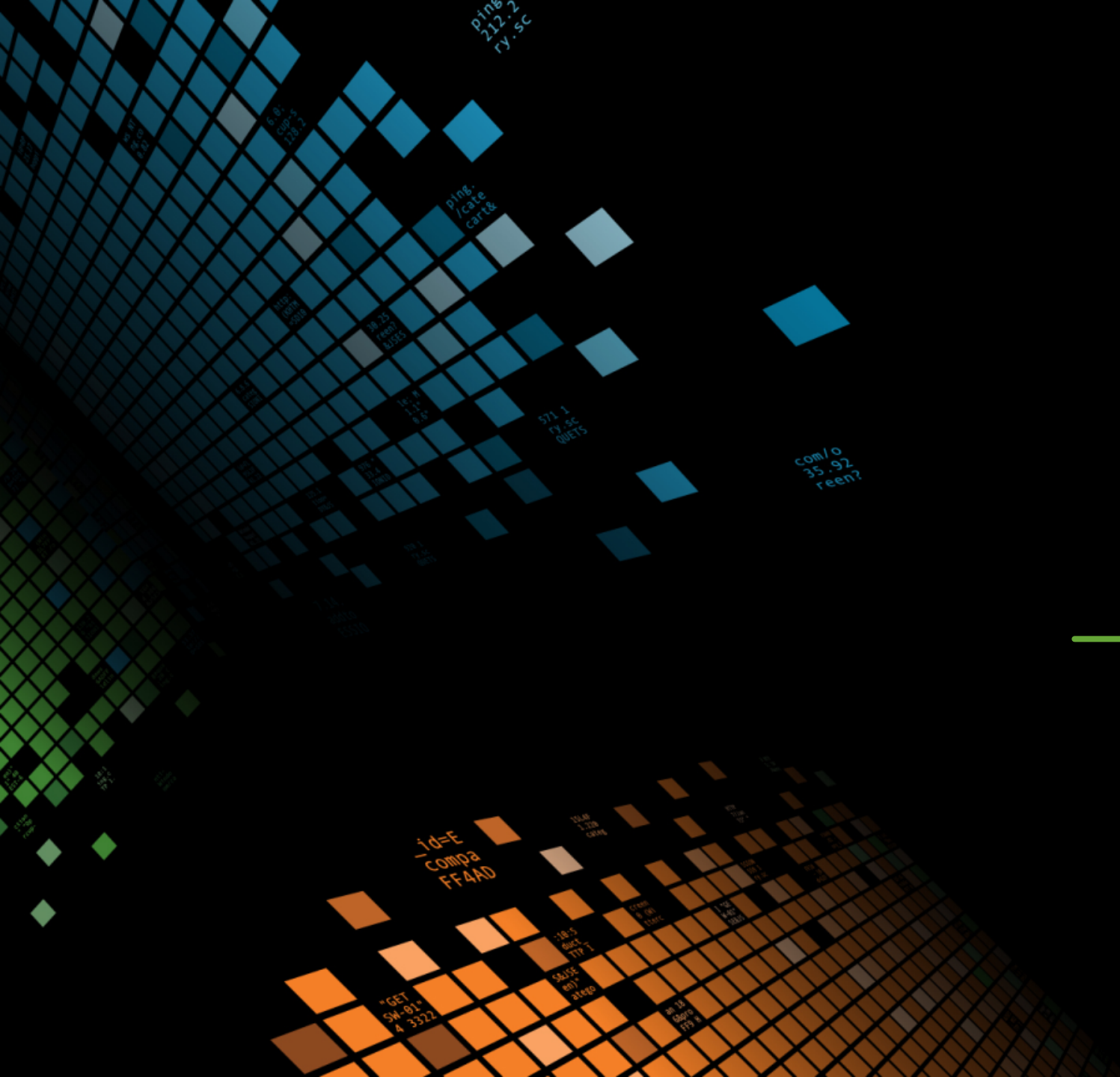
✓ 30 events (before 8/31/17 4:37:35.000 PM)

Original search included: ? ☑ 20 per page ∨ < Prev 1 2 Next >

filter | Apply | Sample: 1,000 events ∨ | All events ∨ | All Events | Matches | Non-Matches

	_raw ⇕	URL ⇕	domain ⇕
✓		http://damtare.by.ru/id.txt	damtare.by.ru
✓		http://www.lowermybills.com/images/home_banner/home_auto_insurance.jpg	www.lowermybills.com
✓		http://www.lowermybills.com/dwr/engine.js	www.lowermybills.com
✓		http://www.lowermybills.com/dwr/interface/RemoteLogger.js	www.lowermybills.com
✓		http://www.lowermybills.com/	www.lowermybills.com
✓		http://damtare.by.ru/id.txt	damtare.by.ru

splunk> .conf2017



Other Notes

Performance Considerations

- Some complex field extractions can be costly
- Some complex regular expressions can be costly
- **Use the Job Inspector** to see if there is a difference in doing on complex field extraction vs man simple field extractions (| rex | **vs** | rex | rex | rex)
- **Sometimes the readability is more important** than the performance

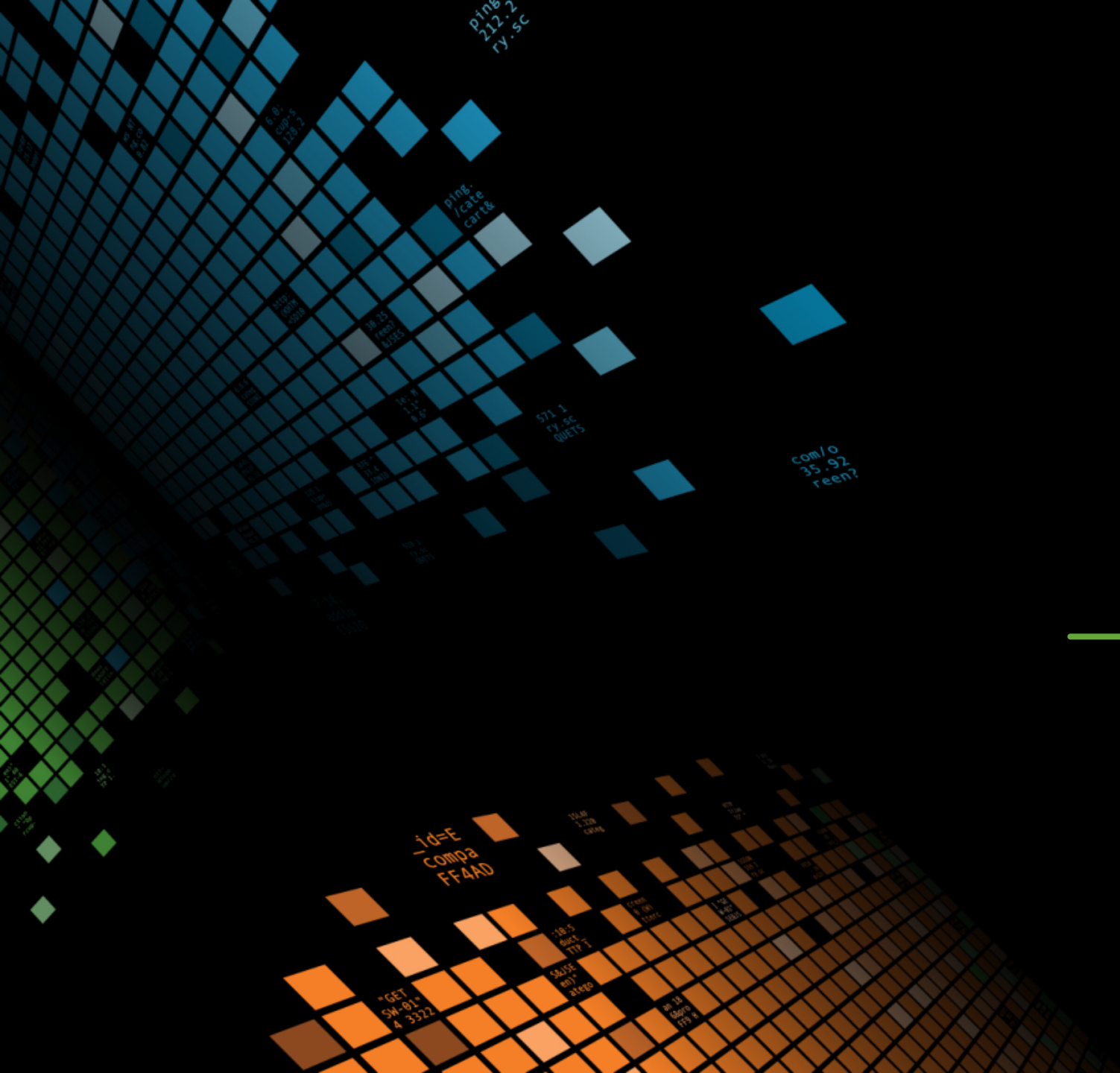
splunk> .conf2017

Maintenance Considerations

- The **complex field extractions** (for example, one that extracts 6 fields at once) **may be easier to maintain** than multiple simple extractions (where you would have 6 different fields extracted by 6 different regexes)

- Your own field extractions will probably be easier to maintain than those created by the Field Extraction Tool – just **write your own regexes better** than the FET

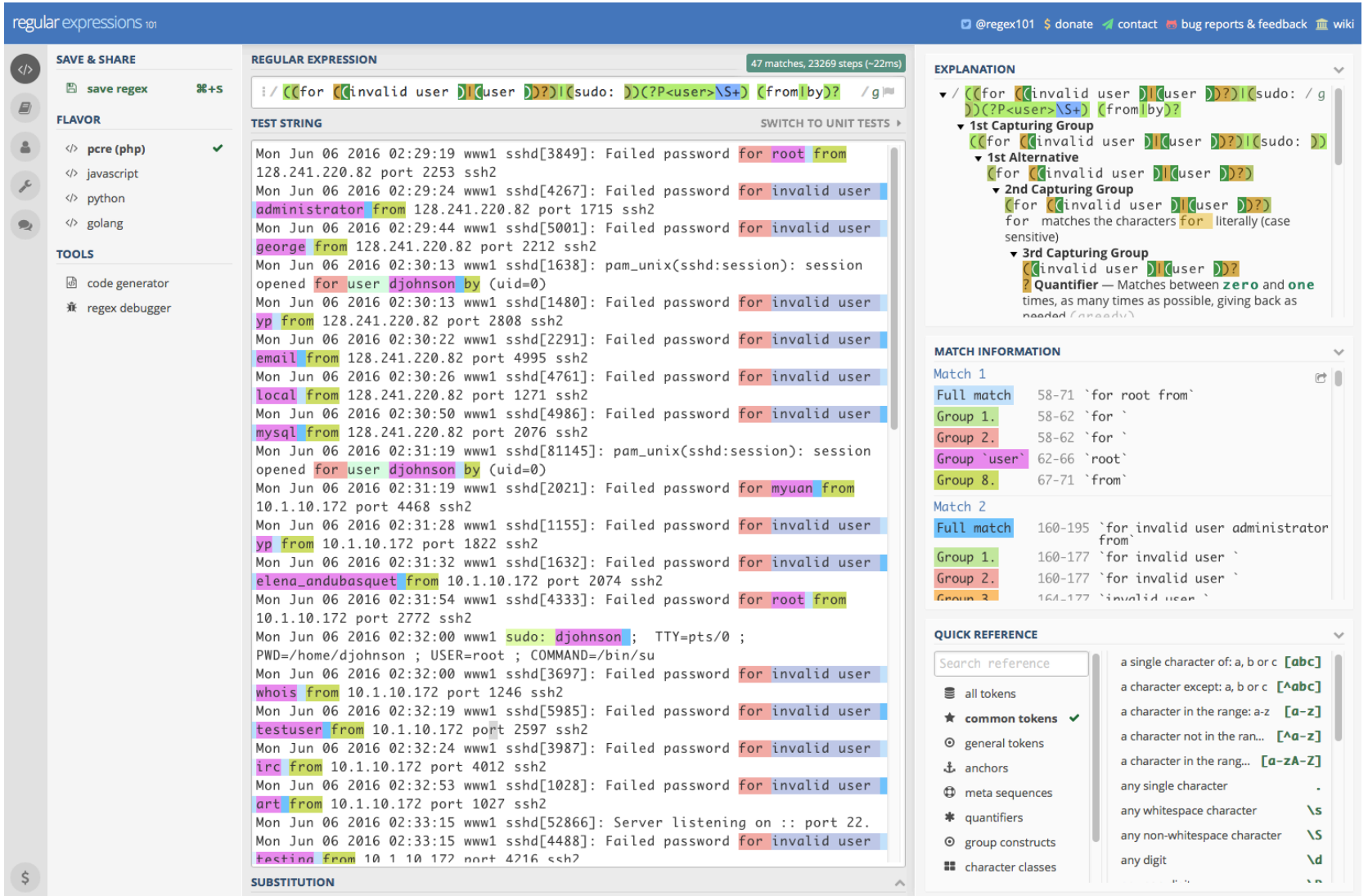
- **Regexes can save you a lot of headaches compared to using non-regex field extractions** (one user was trying to extract data using 200 non-regex evals compared to 6 regexes that accomplished the same thing!)



Tools

Regex101 Web Page

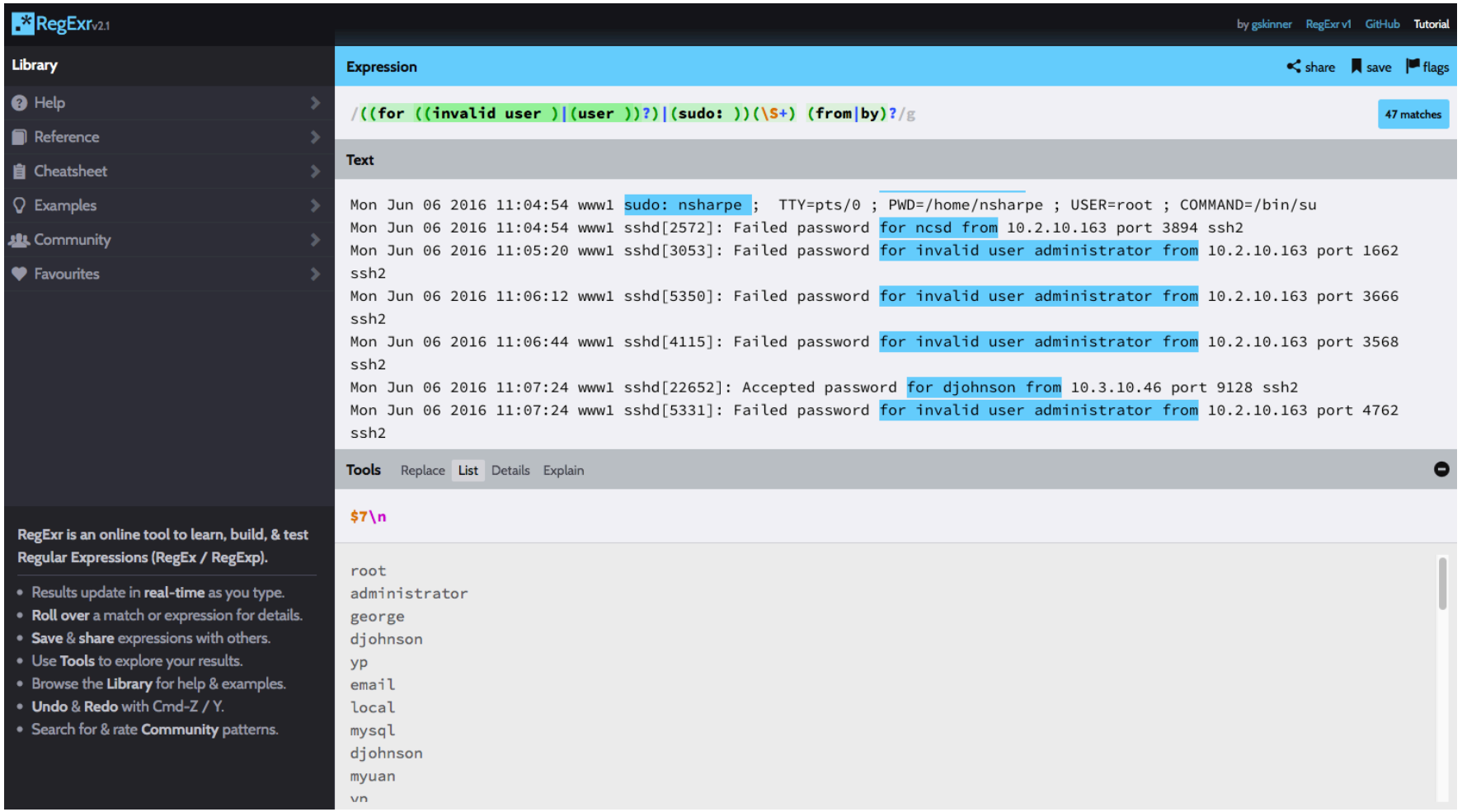
http://regex101.com



splunk> .conf2017

Regexr Web Page

http://regexr.com - Doesn't do PCRE!!



splunk> .conf2017

Improve Regex Performance

- https://www.loggly.com/blog/five-invaluable-techniques-to-improve-regex-performance/

Just for fun:

Try your regex prowess

- Regex Golf
  - https://alf.nu/RegexGolf
  - https://www.oreilly.com/learning/regex-golf-with-peter-norvig

- Regex Crosswords
  - https://regexcrossword.com
  - https://mariolurig.com/crossword/

splunk> .conf2017

Splunk Answers and Docs

Learn from others – ask questions – get answers

- http://answers.splunk.com/

Splunk Documentation

- https://docs.splunk.com/Documentation/Splunk/6.4.3/Knowledge/AboutSplunkregularexpressions

- Splunk **regex** Slack channel

splunk> .conf2017

Acknowledgements

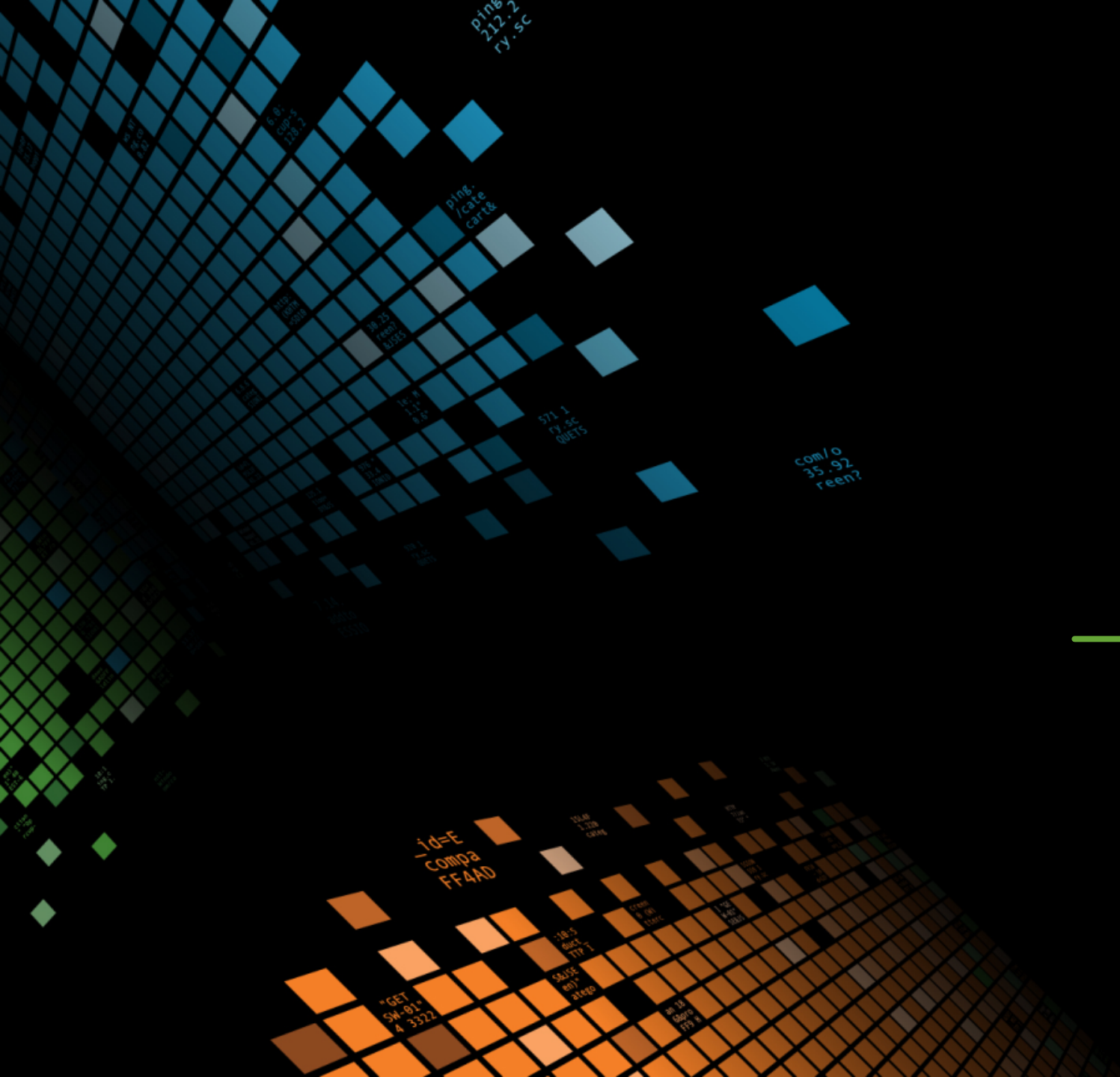
- Lisa Guinn
  - Inspiration and guidance in preparing
  - Data set to use in examples

splunk> .conf2017

THE CHURCH OF

JESUS CHRIST

OF LATTER-DAY SAINTS



Questions?

splunk> .conf2017