Regular Expressions for Incident Response

Sql_Injection=*

Daniel Nutting, Security Operations Manager
Bryan Turner, IT Security Analyst

Publix Super Markets

July 2018 | Version 2.0

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> .conf18

BOTS Tie Ins: Presentations and Hands Ons

- **Hunting the Known Unknowns: Office 365 and Microsoft Azure.** Kovar and Brant. Tuesday 4:45-5:30 PM
- **Splunking the Endpoint IV: A New Hope.** Brodsky. Wednesday 3:15-5:15 PM
- **Hands On with ES, Phantom, and BOTS Data.** Kovar, Smit, Brodsky, Stoner, Valites, Lee. Tuesday 2:15-4:15 PM
- **AWS Security Hands-On.** Valites and Lee. Wednesday 12:45-2:45 PM

Regular Expressions

Introduction

splunk> .conf18

Previous .conf Presentations

- 2016 – Gabriel Vassuer
  Reg(ular expressions?|ex(p|es)?)

- 2016 – Cary Petterborg
  Beyond Regular Expressions

- 2017 – Michael Simko
  Regex in your SPL
  - GREAT INTRODUCTION!

- RegexOne.com

splunk> .conf18

Agenda

- Prerequisites
- RegEx Philosophy
- Develop RegEx* Faster
- Common Use Cases
- Work Around Parsing Errors
- SQL Injection example
- XSS example
- Sysmon example

Pre-requisites

Comfort with Regex

- `\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}`
  IP Address

- `(?<email>.+@.+)\s`
  Field extraction

- `\s\S\d\D+\w\W?.*`

- **`regex|rex`** *vs* **`r(?:eg)?ex`**

splunk> .conf18

Shift some Paradigms!

- Post Alert -> Incident Response & Investigations
- Speed Over Precision
  - Easy to understand is better than plug and play
- Accuracy Over Perfection
  - False positives are okay
  - False negatives are not okay

Validating RFC822 Email Addresses

From Stack Overflow:

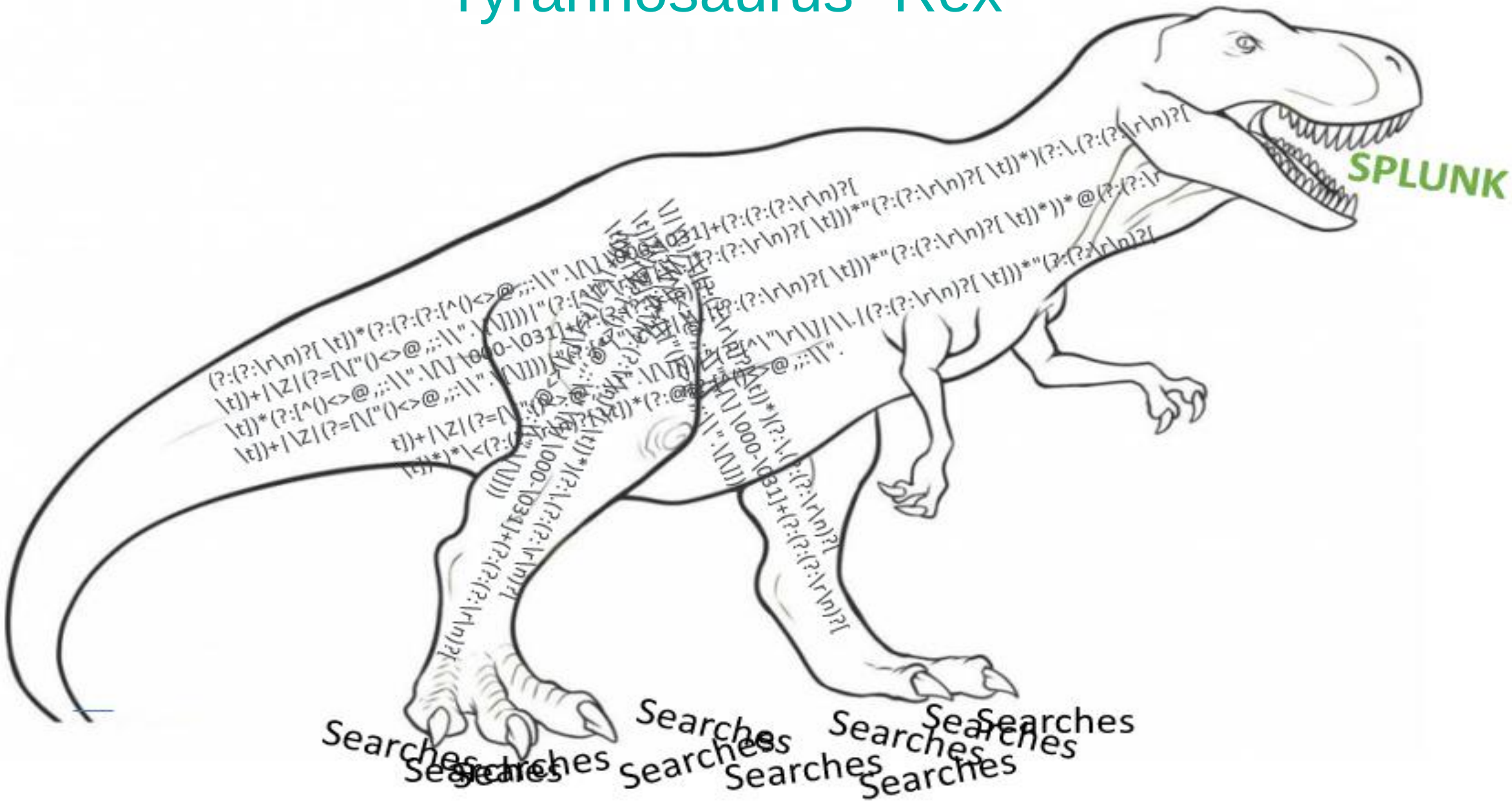
```
(?:(?:\r\n)?[ \t])*(?:(?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[
\t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[
\t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[
\t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[
\t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[
\t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[
\t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[
\t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[
\t])*))*|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[
\t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[
\t])*)*\<(?:(?:\r\n)?[ \t])*(?:@(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[
\t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[
\t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[
\t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[
\t])*))*(?:,@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[
\t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\
```

- Roughly 20% of actual regex

splunk> .conf18


YOU WERE SO PREOCCUPIED WITH WHETHER OR NOT YOU COULD
YOU DIDN'T STOP TO THINK IF YOU SHOULD

Tyrannosaurus "Rex"



Example

Speed over Precision

- Which is easier to understand?
  - `([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})`
  - `.+@.+\..+`
- Just worry about getting the data

Example

Accuracy over Perfection

- Which is easier to understand?
  - `([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})`
  - `.+@contoso.com`
- Just worry about getting the data

splunk> .conf18

Use Case: IPv4 Addresses

Only worry about accuracy if you have to

- 192.168.1.1
- 10.10.10.3
- **1.800.555.5555**
- 8.8.8.8
- 10.10.10.5

- `[\d\.]+`
- `(\d{1,3}\.){3}\d{1,3}`

Stack Overflow's suggestion:

```
\b((?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)(?:(?<!\.)\b|\.)){4}
```

Develop RegEx* Faster

***Good Enough RegEx**

splunk> .conf18

Process

1. Identify potential anchors
   - `http://www.example.com/sample/url`
   - `Jane.Doe@example.com`
2. Set up exclusions
   - `| rex field=url “http:\/\/(?<domain>.+[^\/]”`
3. Set boundaries on quantifiers
   - `| rex field=url “http:\/\/(?<domain>.{100}[^\/]”`

Think Finite

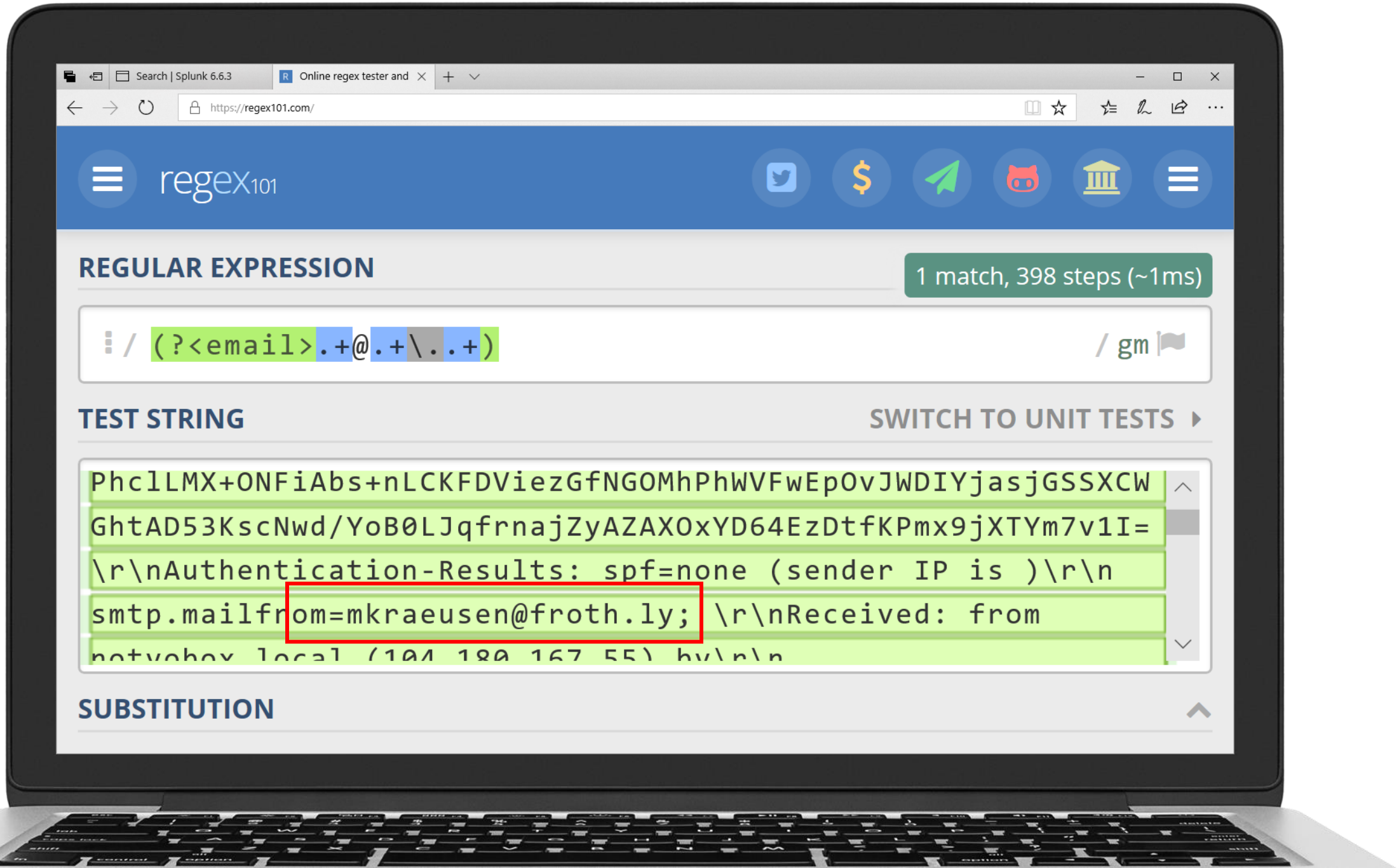
- Get all the MATCHES?

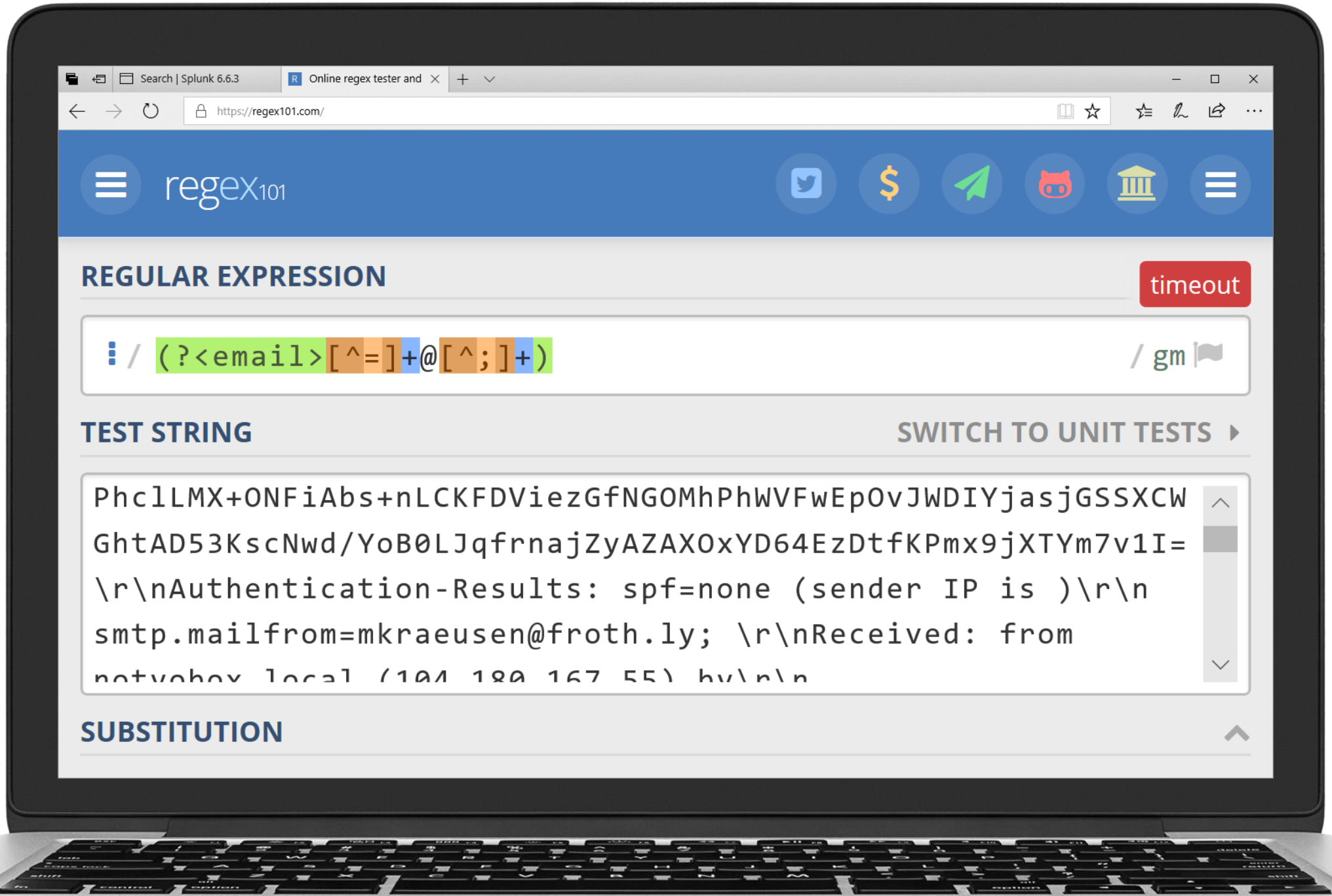
```
[a-z]+
```

- Let's be realistic:

```
[a-z]{1,64}
```

- Easier on the Regex Engine
  - Fewer Steps = Faster Search





Focus on the Negative

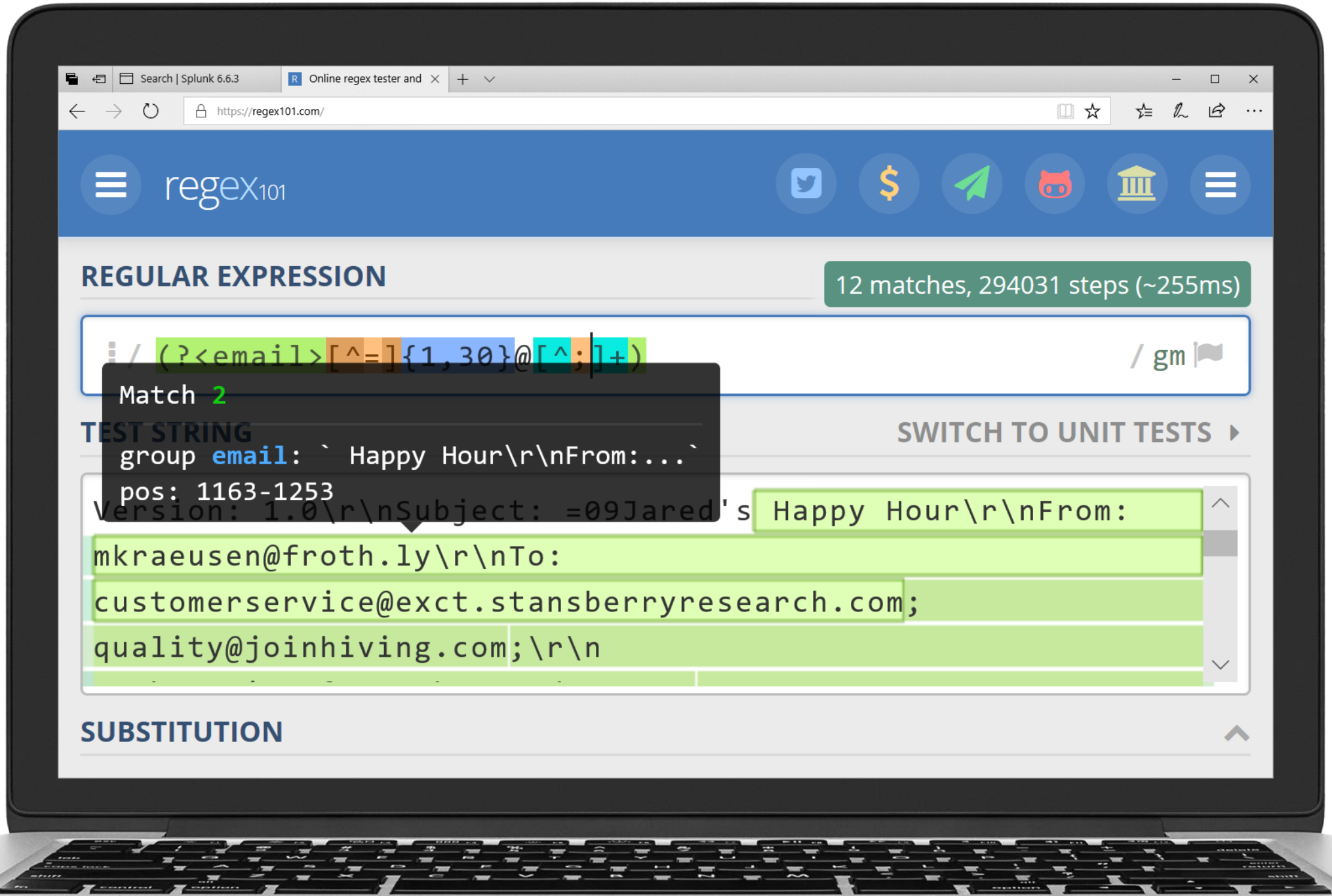
- Inclusive?

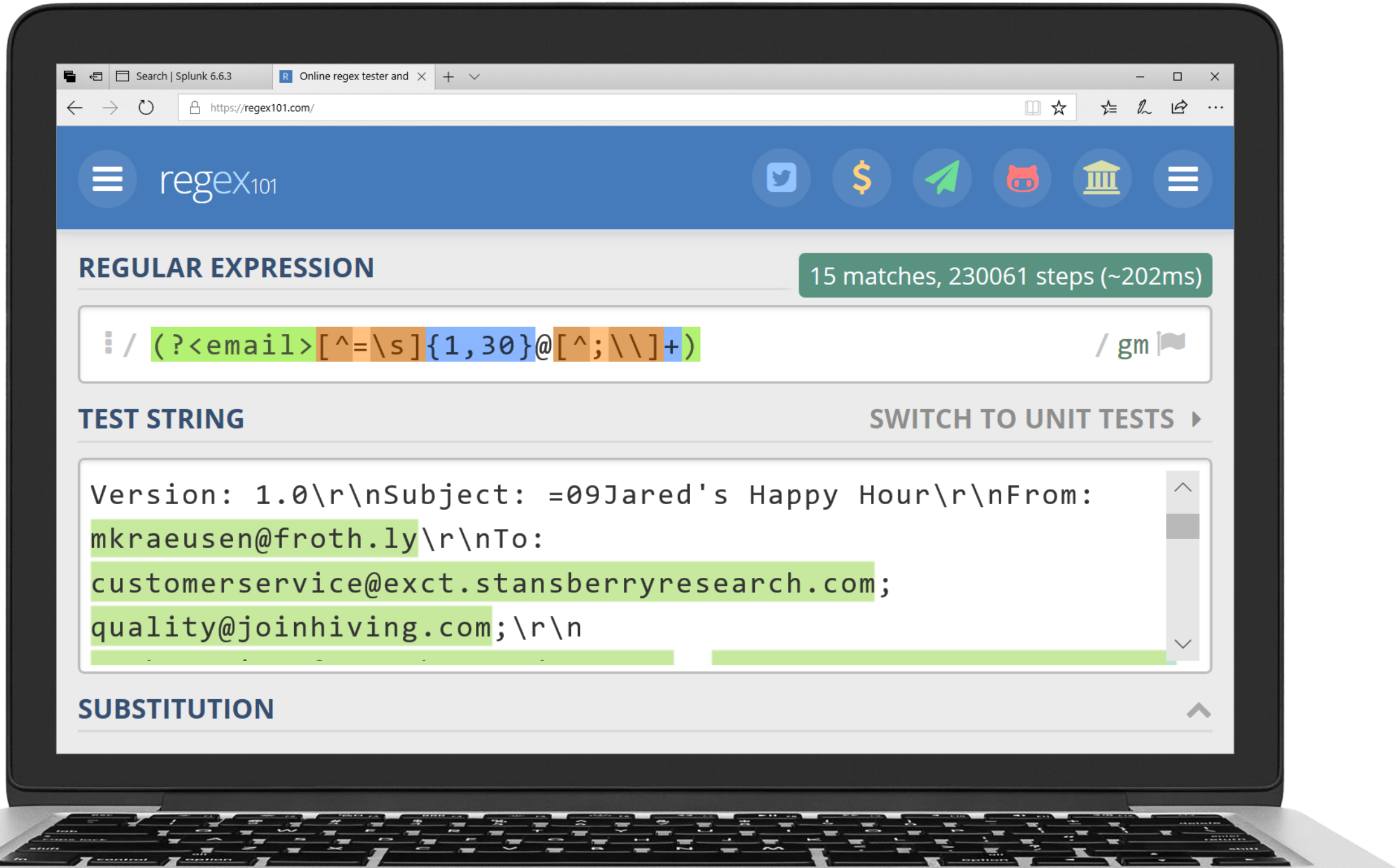
```
([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})
```

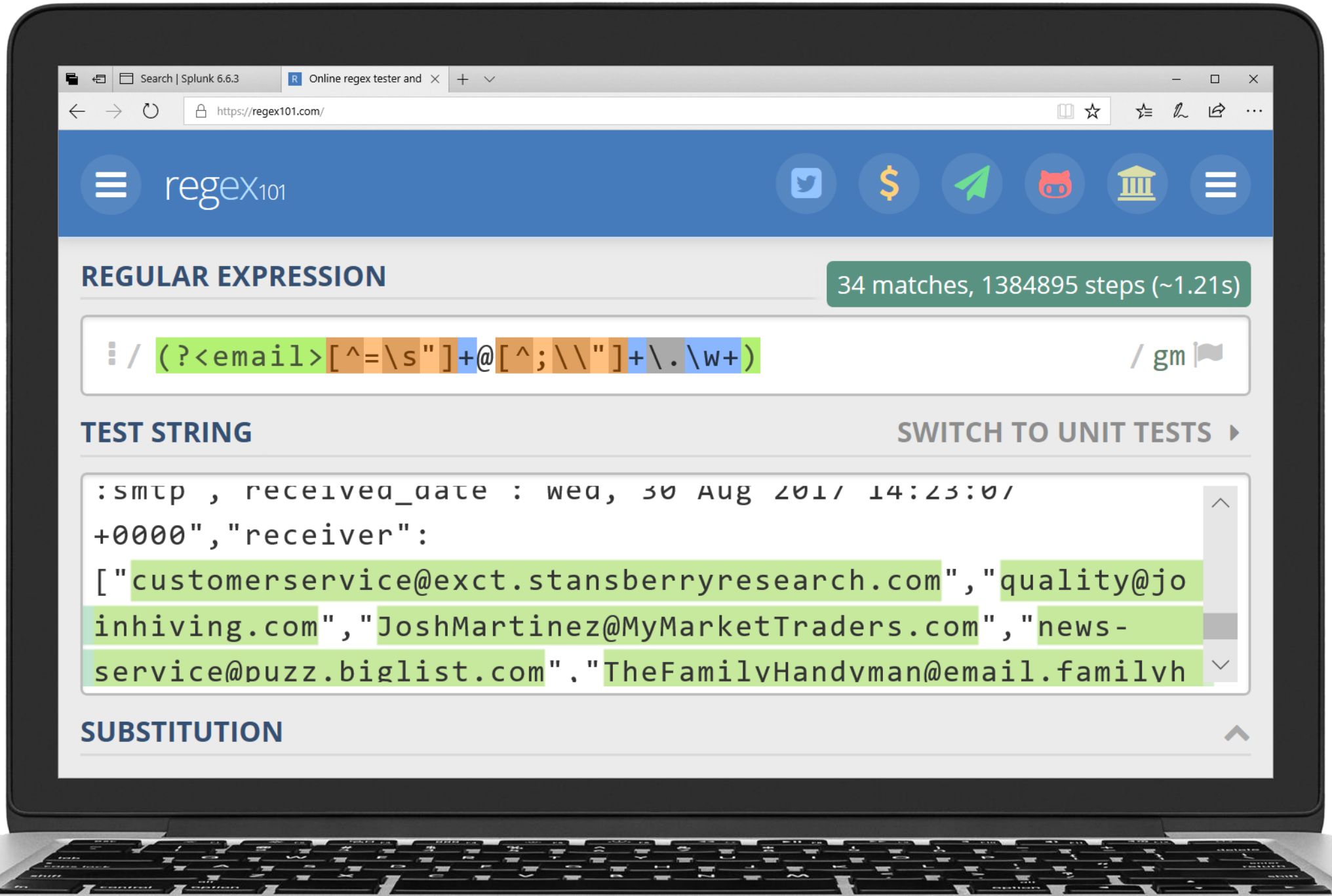
- Exclusive!

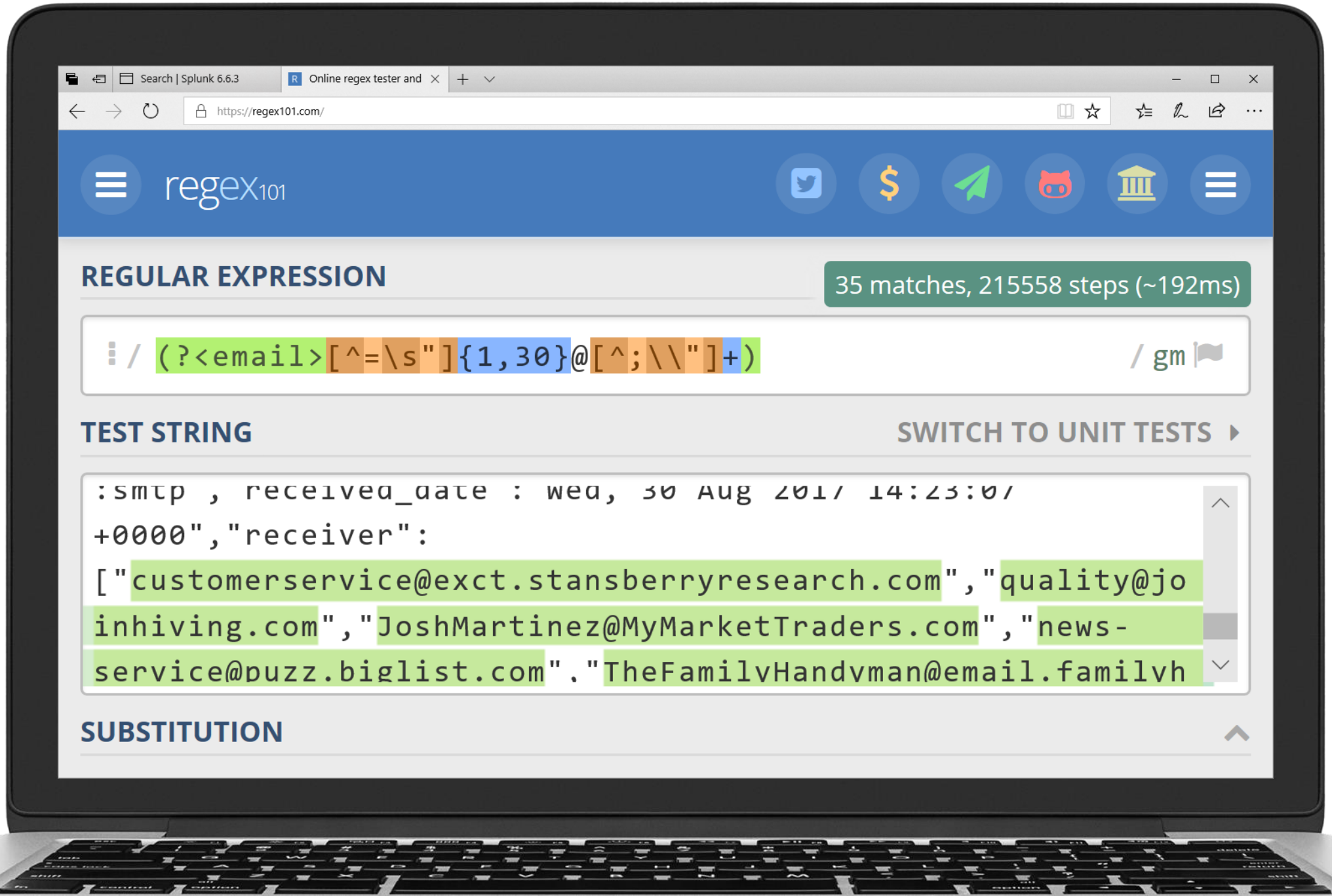
```
[^\s]+@[^\s]+\.[^\s]+
```

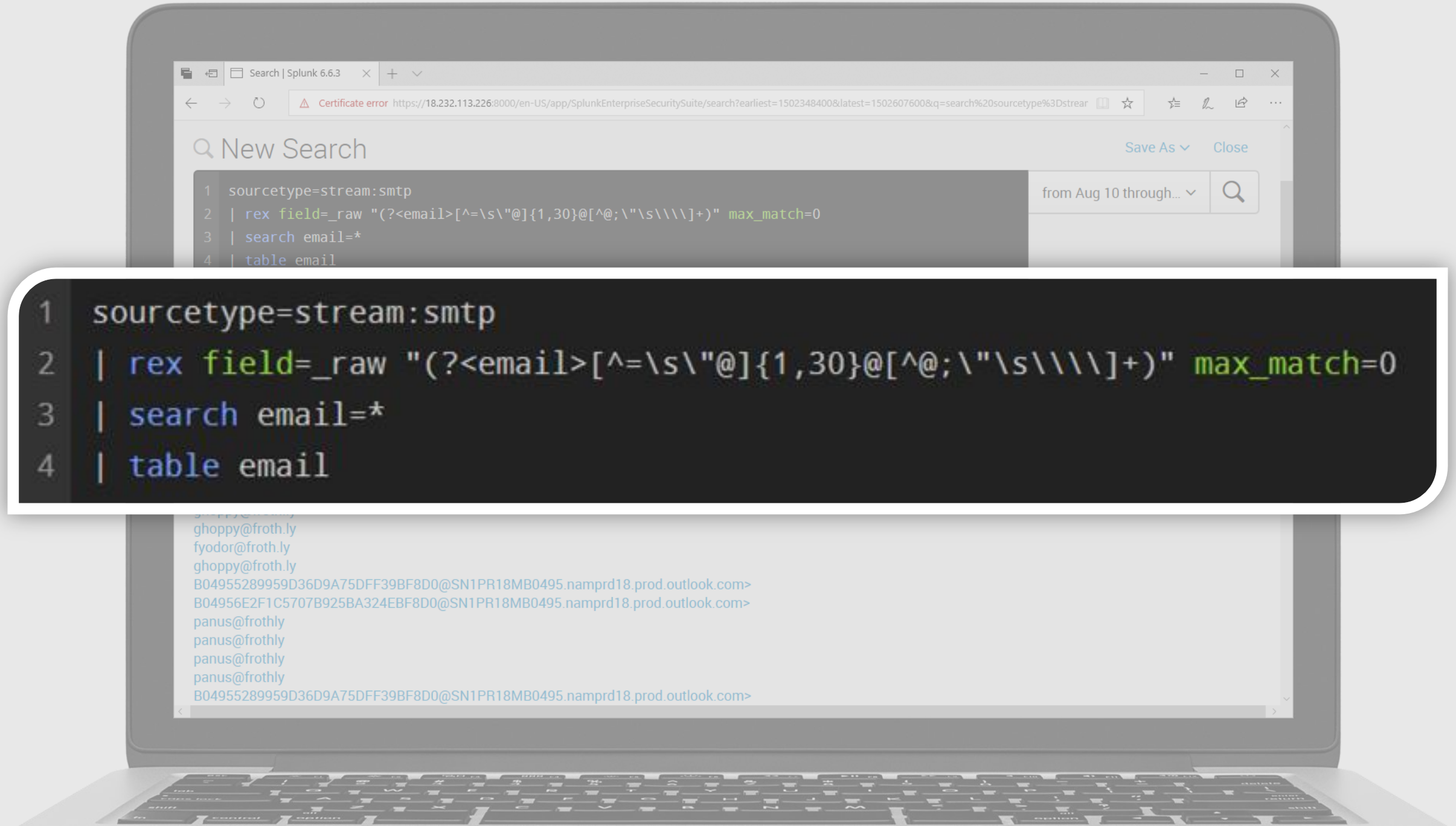
- Faster to add bad characters as you test.












```
1 sourcetype=stream:smtp
2 | rex field=_raw "(?<email>[^=\s\"@]{1,30}@[^@;\"\s\\\\]+)" max_match=0
3 | search email=*
4 | table email
```

Break It Down

```
(?<email>[^=\s\"@]{1,30}@[^@;\"\s\\\\]+)
```

- Capture Group
- Character Class for Mailbox Name
- Quantifier (Repeats 1 to 30 times)
- Literal “@” sign
- Character Class for Domain Name

Break It Down

```
(?<email>[^=\s\"@]{1,30}@[^@;\"\s\\\\]+)
```

- Weird Splunk thing:

  If you want “\\” double it

Other Pitfalls

Can Slow Down Your Search:

- Catastrophic Backtracking
- Repeating a Capturing Group

See https://regular-expressions.info

- Assuming All Results are Captured

Common Use Cases

Frequently used searches during investigations

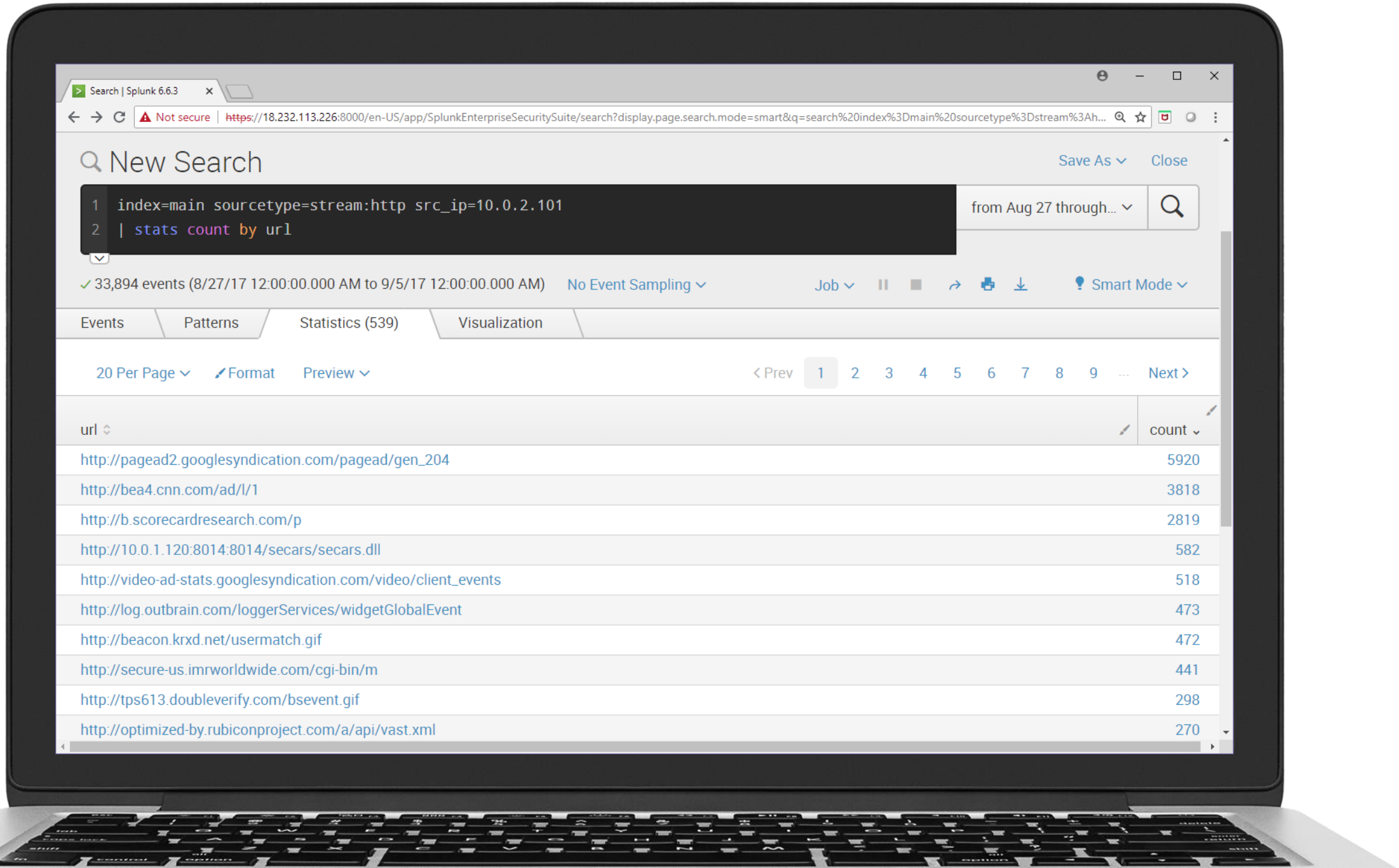
User Activity

- What sites were visited around the time of the compromise?
  - Does anything look suspicious?
- What does the search history look like?
  - Do searches appear to be unrelated to their job?
- Is there any abnormal activity?

Web Browsing

- Do any sites look suspicious?
  - Uncommon top level domains
    - .top, .download, .date
  - Abnormal amounts of traffic
    - Large spikes in activity to unfamiliar sites
  - Potentially concerning in context
    - Competitors
    - File Sharing (Google Drive, Dropbox)

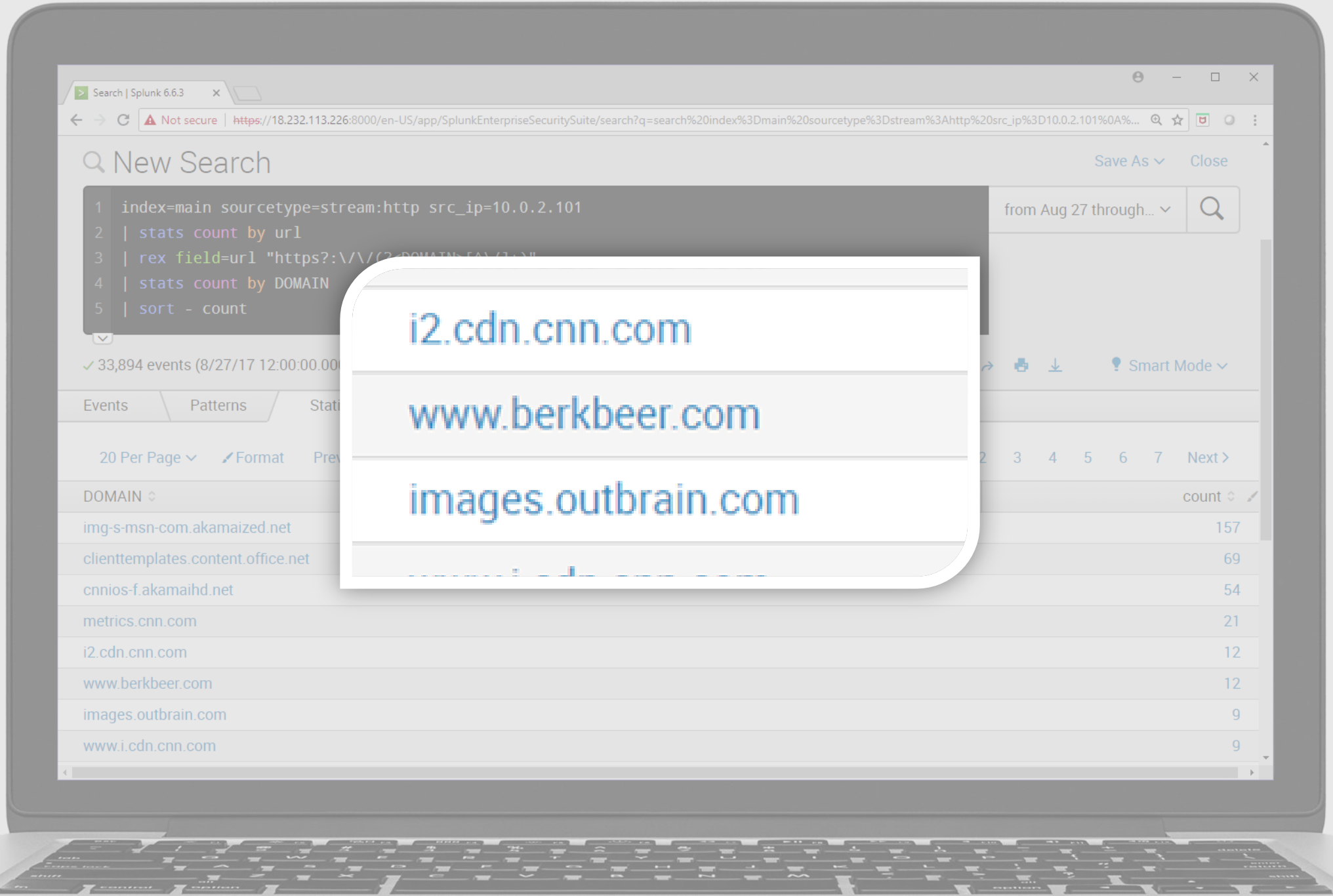
Example: Employee at Frothly Beer Company visited a competitor's website. What was that website?



Domain Search

http://metrics.cnn.com/b/ss/cnn-adbp-domestic/1/H.26.1/s95879181409321

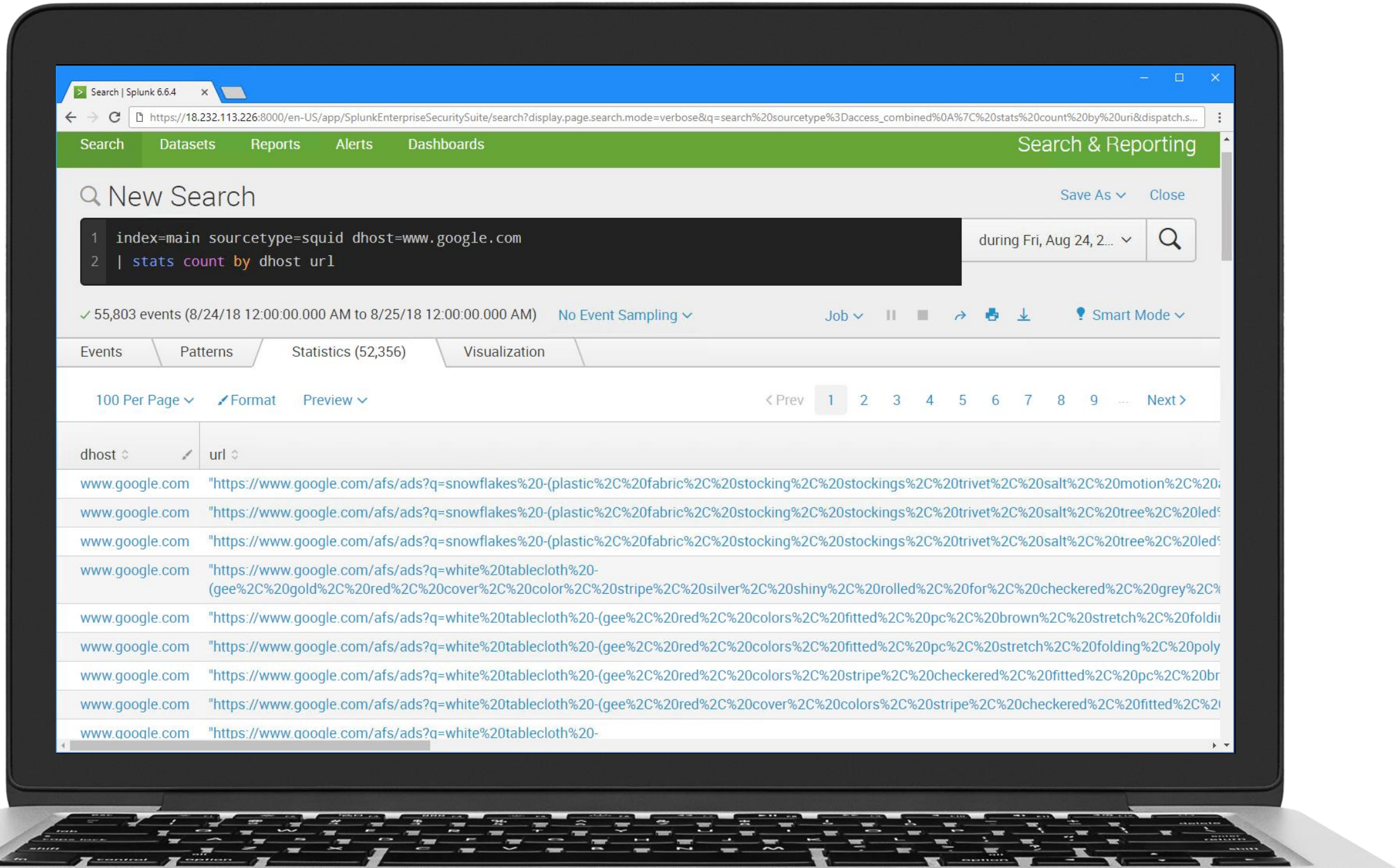
```
index=main sourcetype=stream:http src_ip=10.0.2.101
| stats count by url
| rex field=url "https?:\/\/(?<DOMAIN>[^\/]+)"
| stats count by DOMAIN
| sort - count
```



Search History

Reviewing a user's search history can assist in telling the "story".

- What was the user doing around the time of the alert/compromise?
- What was the user's intent?
  - Were they being careless or malicious?
- Do their searches appear unusual?
  - Are searches related to job activities?

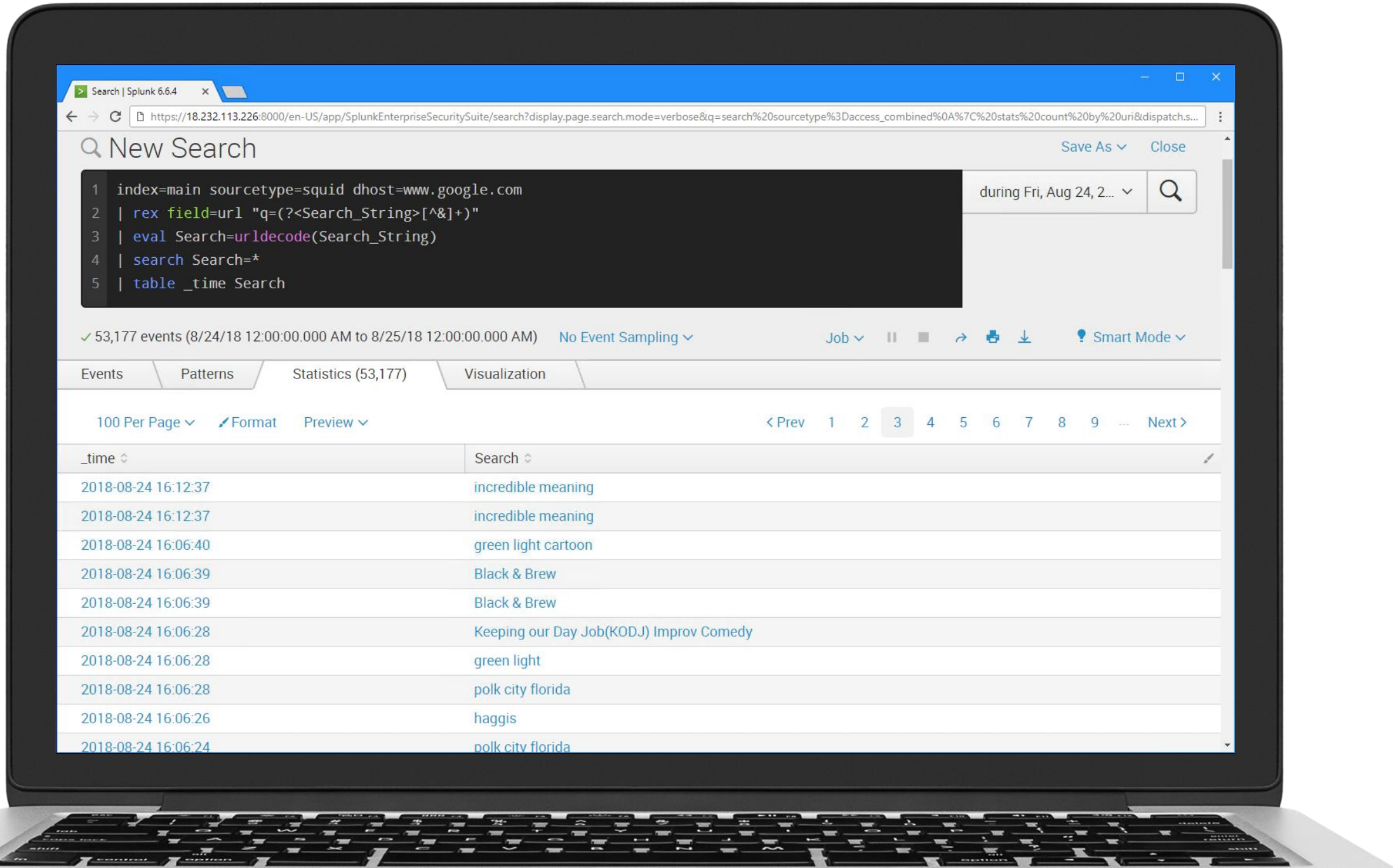


Google Search

Google search queries begin with "q=" and end with "&".

```
Bg&pq=was%20that%20t-rex%20joke%20cringeworthy&cp=
```

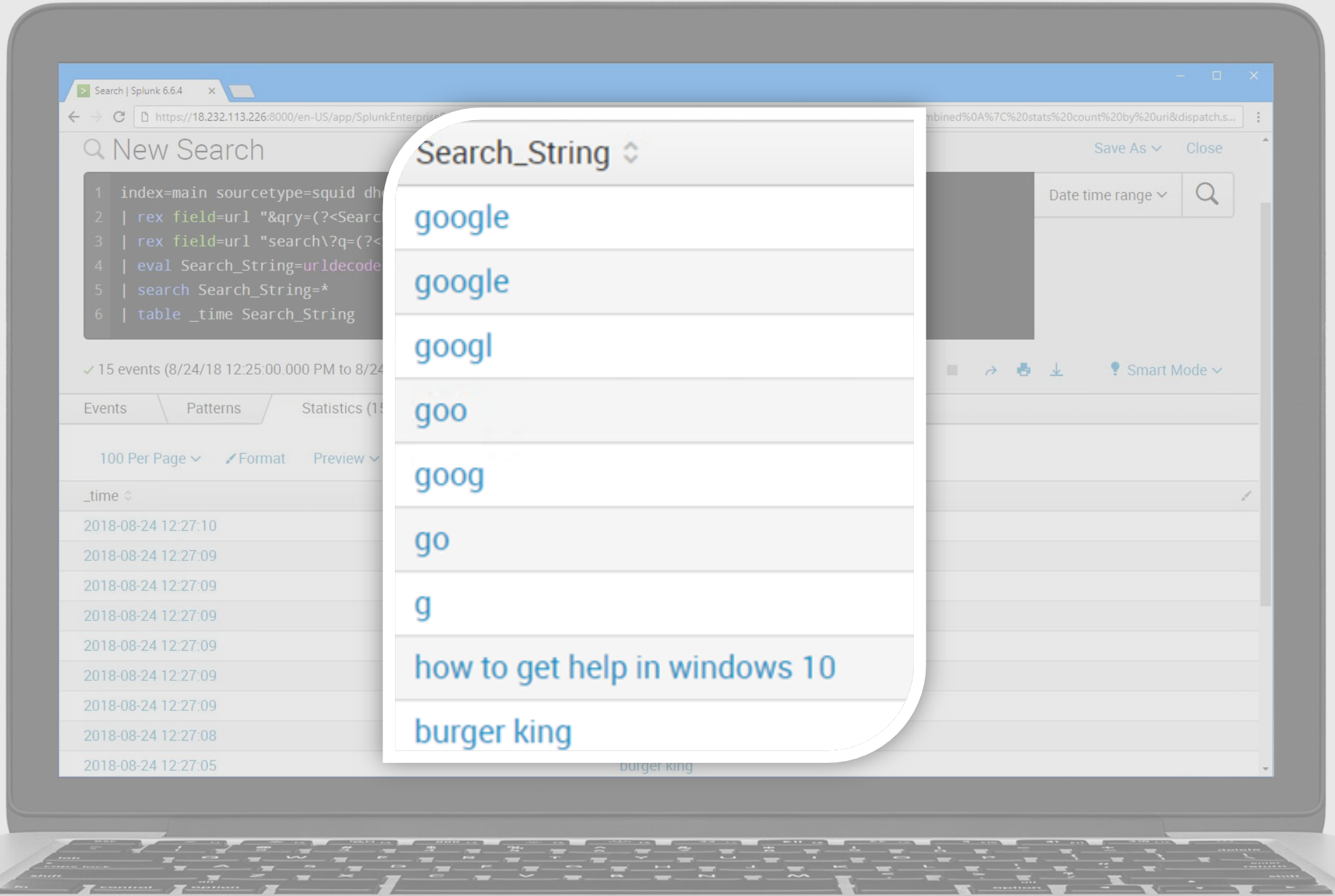
```
index=main sourcetype=squid dhost=www.google.com
| rex field=url "q=(?<Search_String>[^&]+)"
| eval Search=urldecode(Search_String)
| search Search=*
| table Search
```



Bing Search

```
index=main sourcetype=squid dhost=www.bing.com
| rex field=url "&qry=(?<Search_String>[^&]+)"
| rex field=url "search\?q=(?<Search_String>[^&]+)"
| eval Search_String=urldecode(Search_String)
| search Search_String=*
| table Search_String
```

*Note that Bing has two possible search strings.
Be careful not to assume that "results" = "all results"



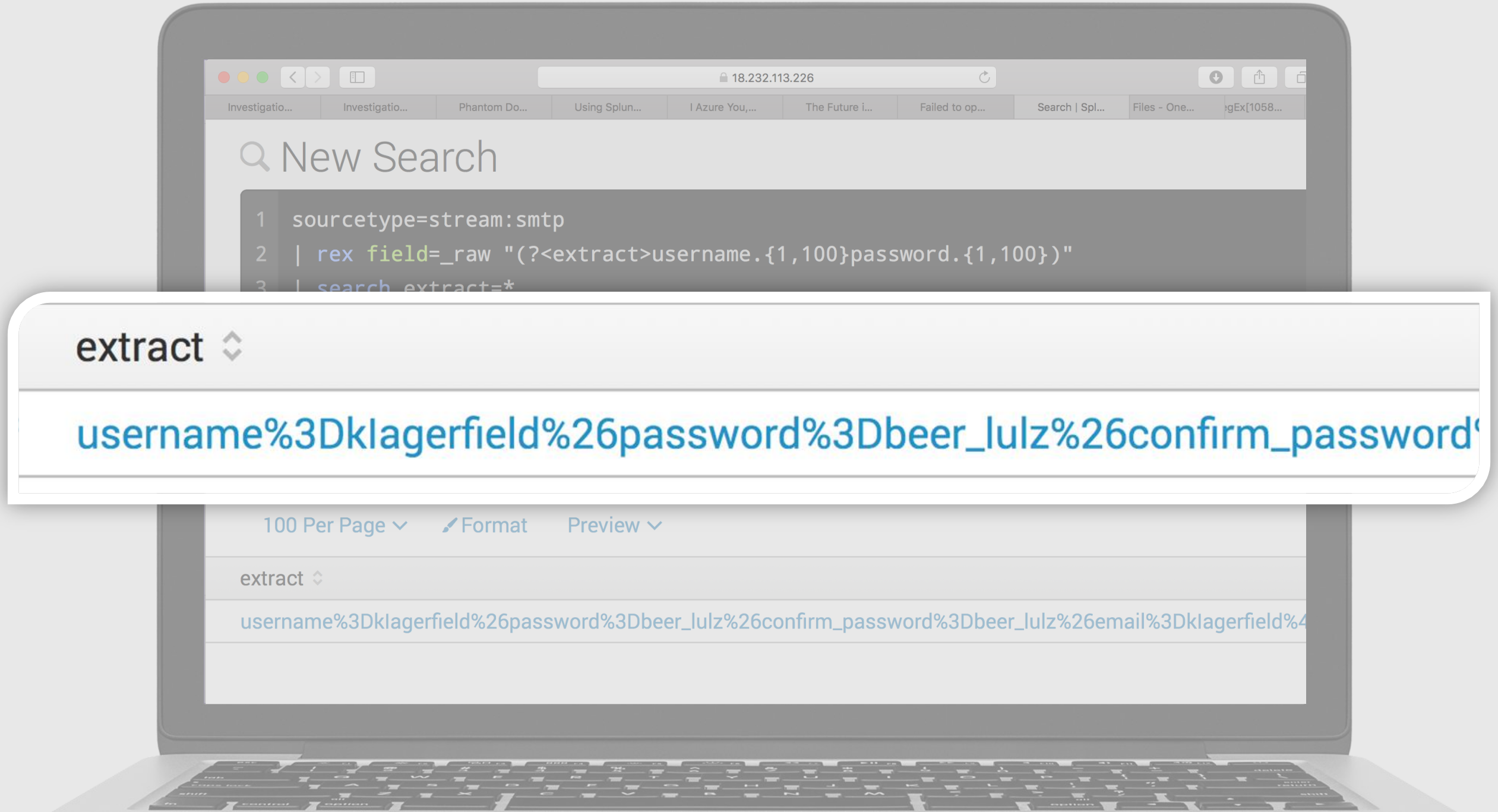
Words near other words

- Think commonly paired words
- Can assist in identifying potential security concerns.

  - Select, From, Where
  - <script>,</script>
  - Download, docm

Example: username, password

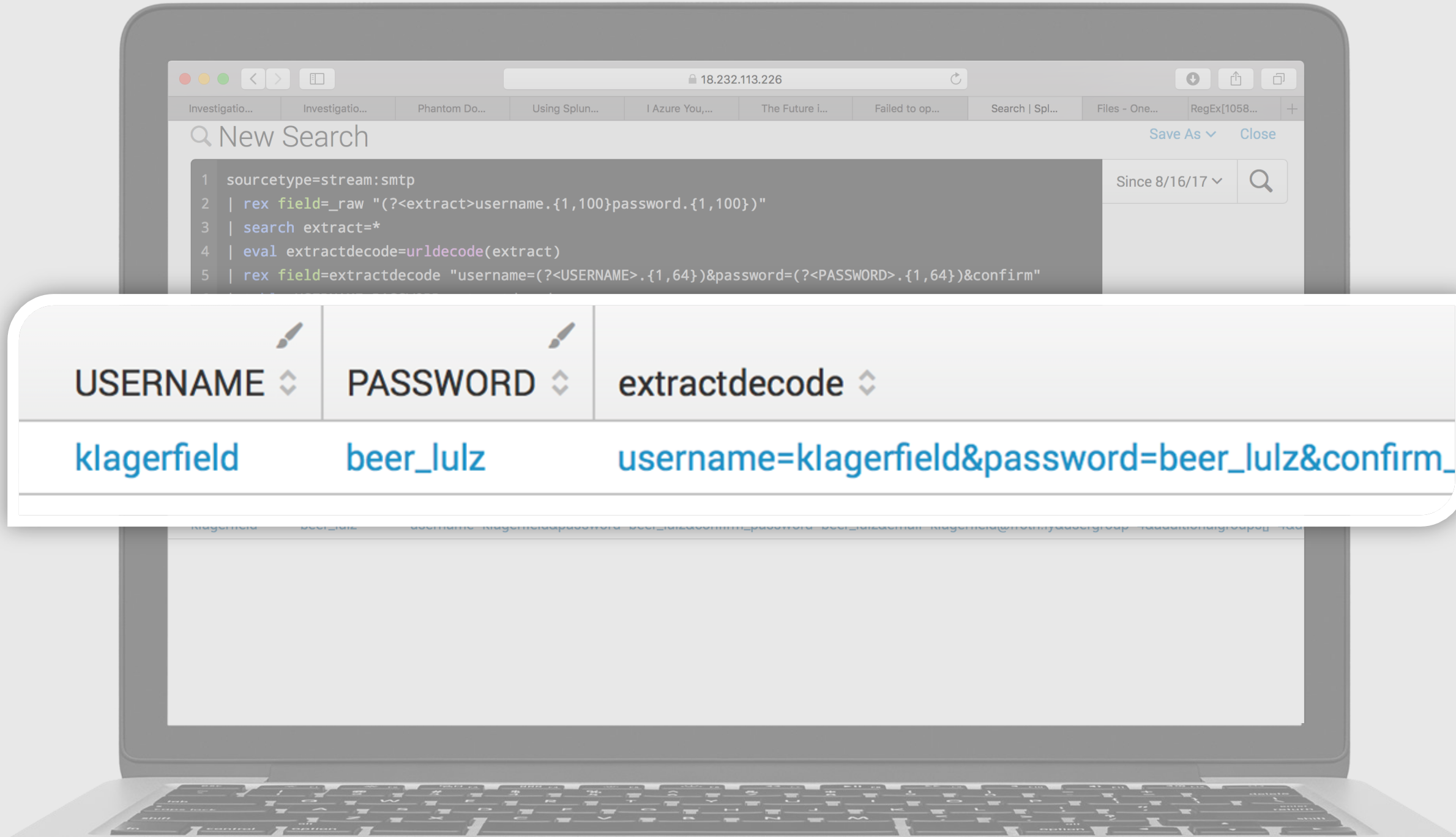
```
| rex field=_raw
"(?<extract>username.{1,100}password.{1,100})"
```

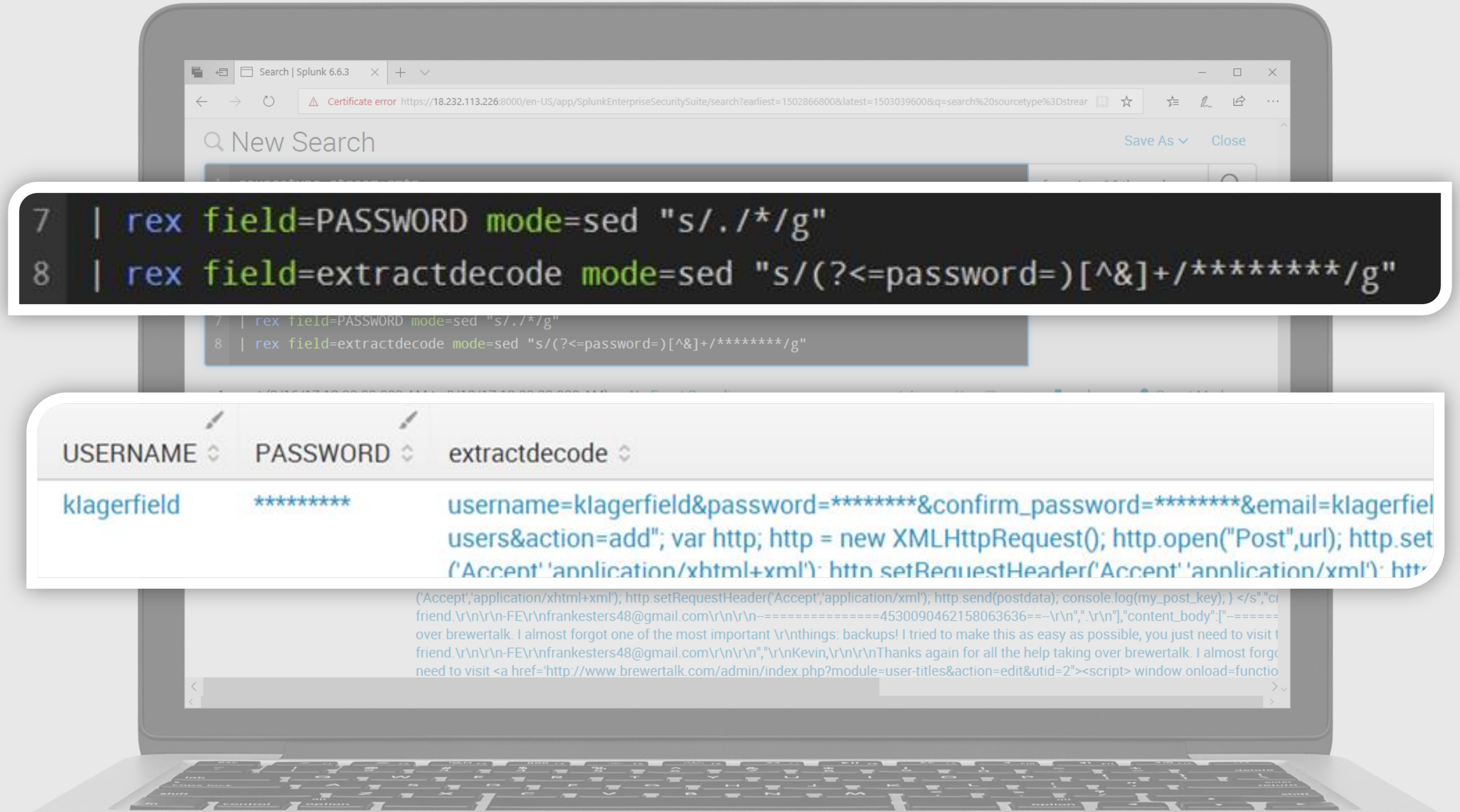


Words near other words

```
sourcetype=stream:smtp
| rex field=_raw
 "(?<extract>username.{1,100}password.{1,100})"
| search extract=*
| eval extractdecode=urldecode(extract)
| rex field=extractdecode
 "username=(?<USERNAME>.{1,64})&password=(?<PASSWOR
 D>.{1,64})&confirm"
| table USERNAME PASSWORD extractdecode
```

splunk> .conf18

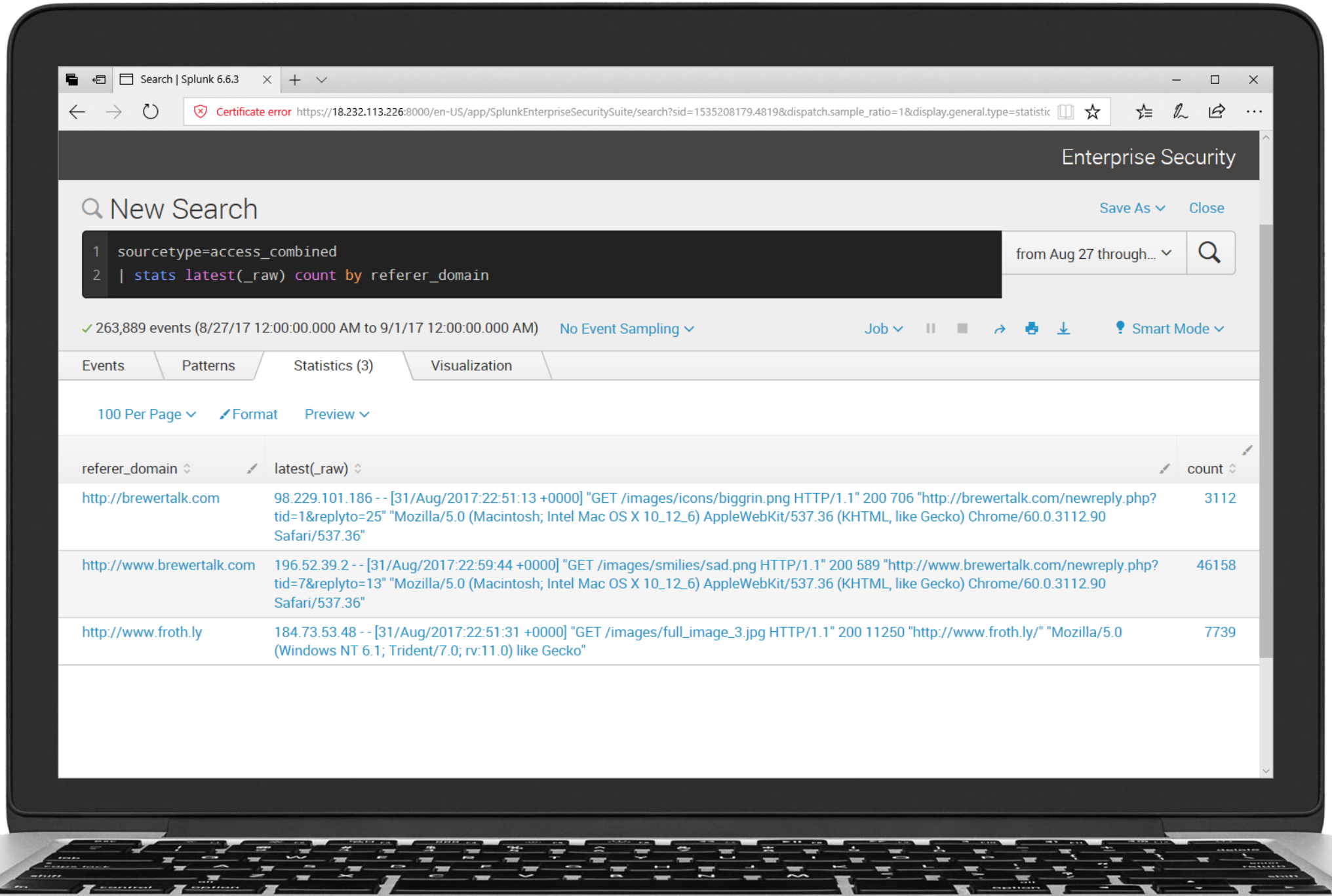


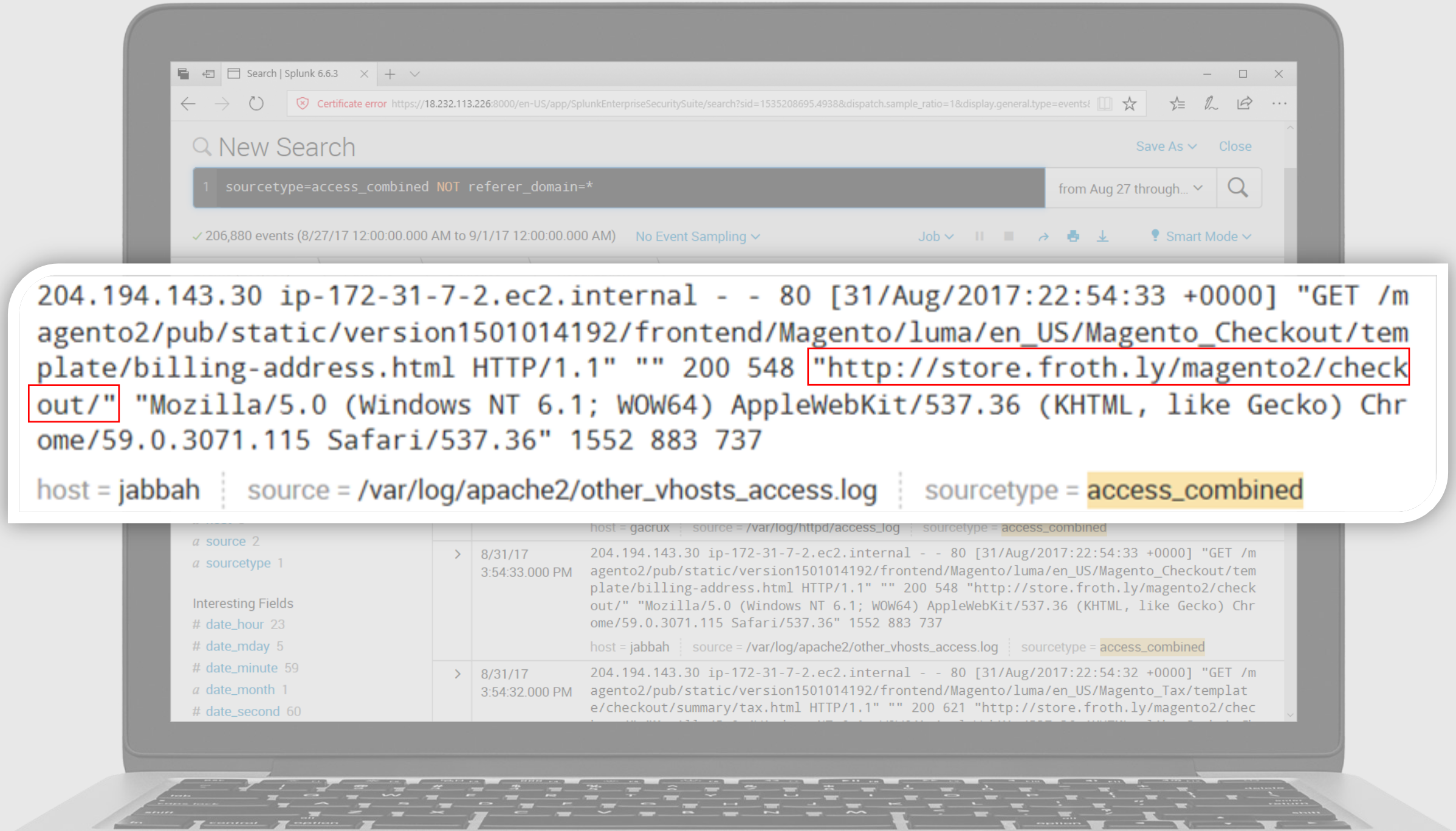


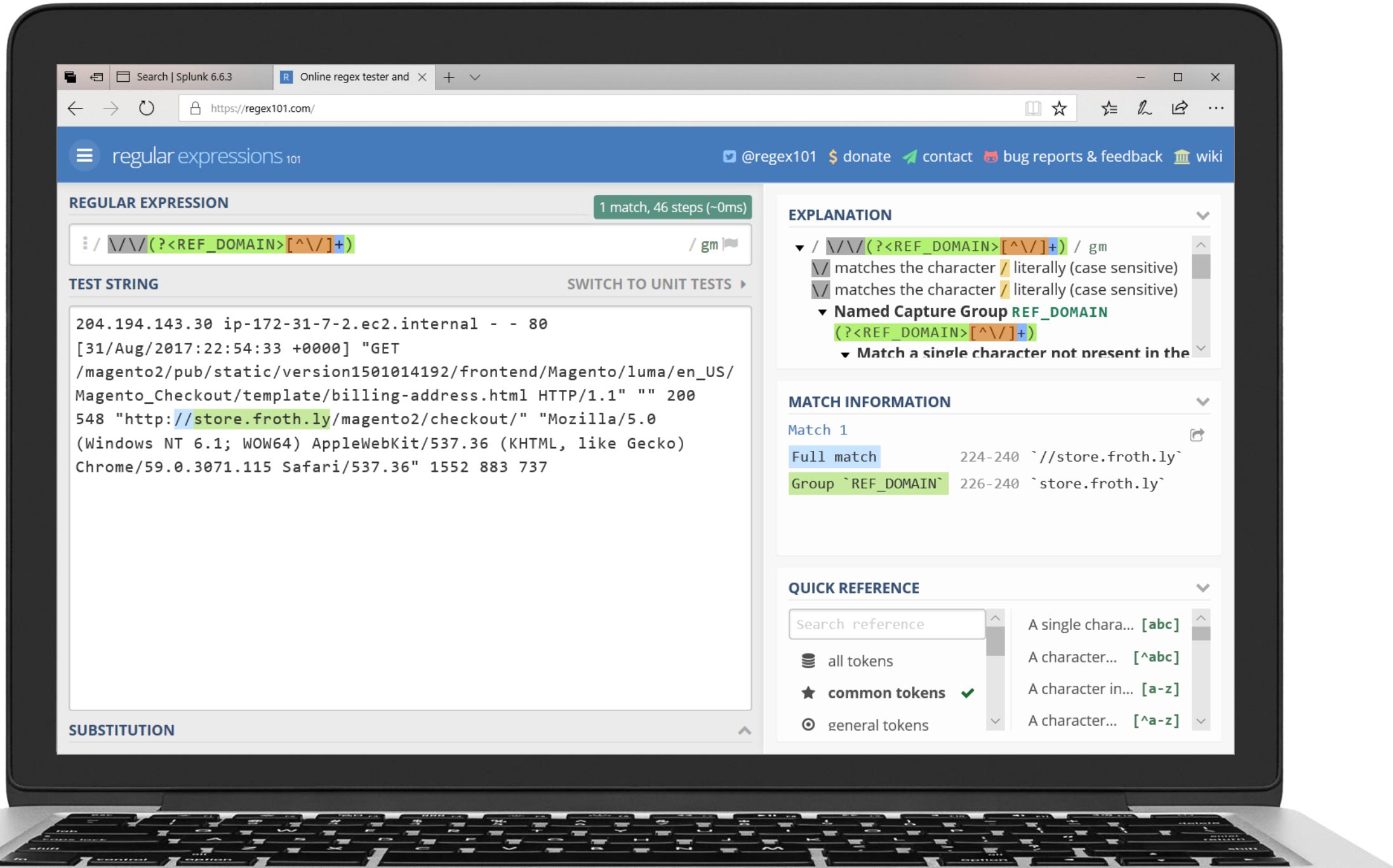
Work Around Parsing Errors

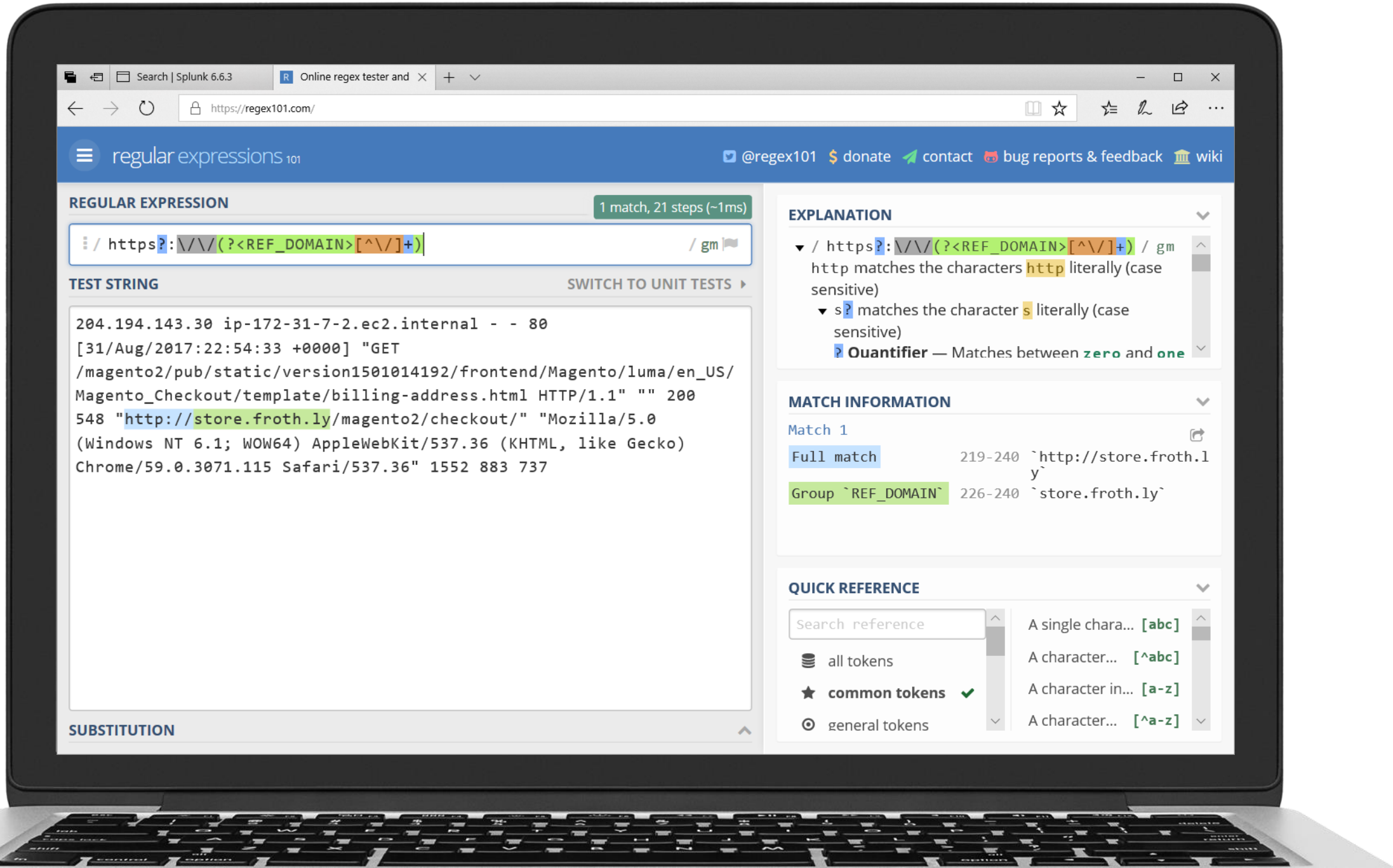
Incident Response

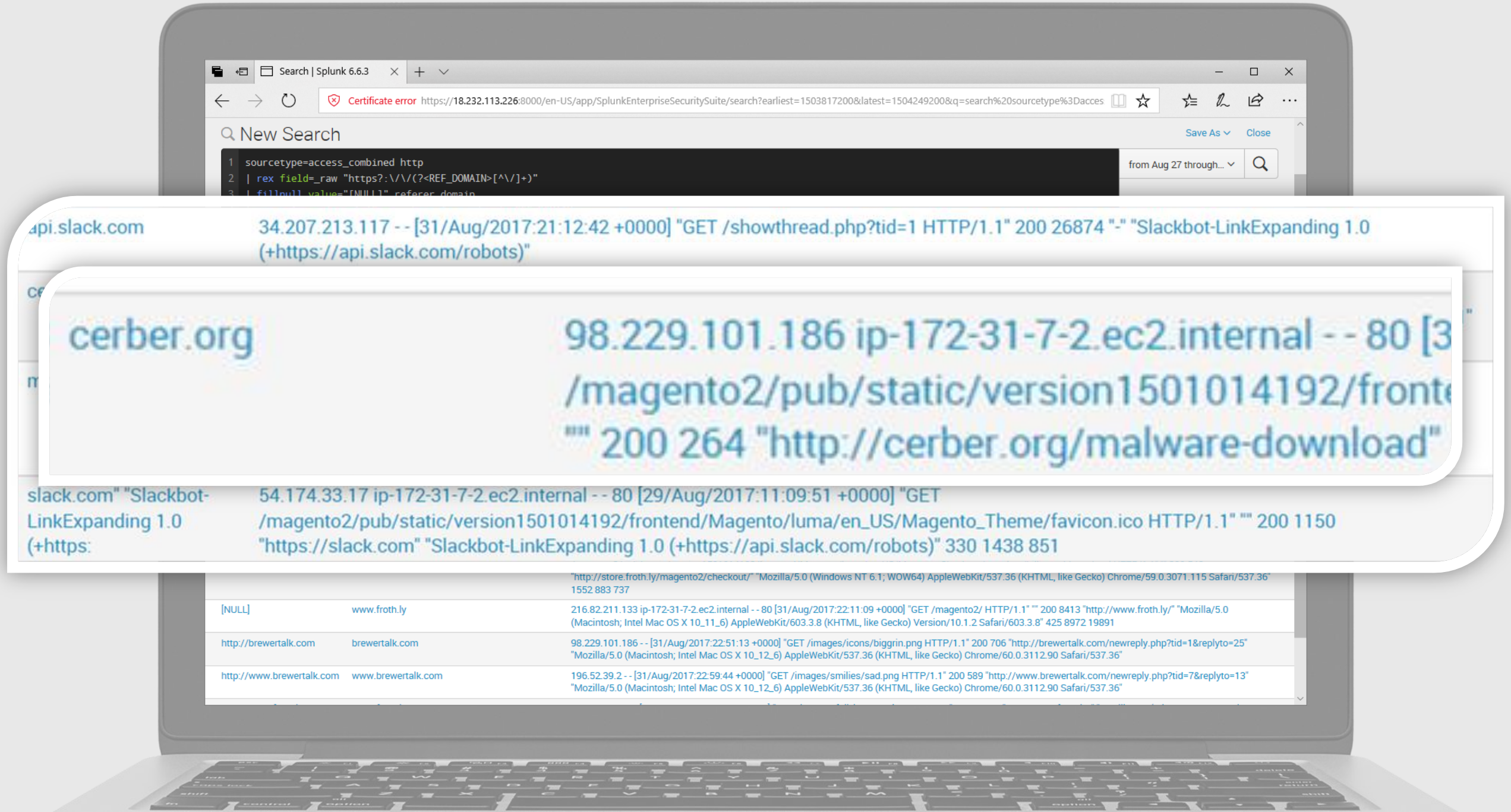
splunk> .conf18











SQL Injection

Incident Response

splunk> .conf18

SQL Injection

According to OWASP, "A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application.“

https://www.owasp.org/index.php/SQL_injection

SQL Injection

- Investigations start somewhere
  - Alert
  - Pen Test Results
- Alert
  - e.g. IDS event indicating a SQL Injection
  - Direct us to a specific log

SQL Injection REGEX - Data

<dt>Query:</dt>

<dd> SELECT q.*, s.sid FROM mybb_questionsessions s

LEFT JOIN mybb_questions q ON (q.qid=s.qid)

WHERE q.active='1' AND s.sid='makman' and

updatexml(NULL,concat (0x3a,(

SELECT email FROM mybb_users ORDER BY UID LIMIT 5,1)),NULL) and '1' </dd>

<dd>1105 - XPATH syntax error: ':**klagerfield@froth.ly**'</dd>

splunk> .conf18

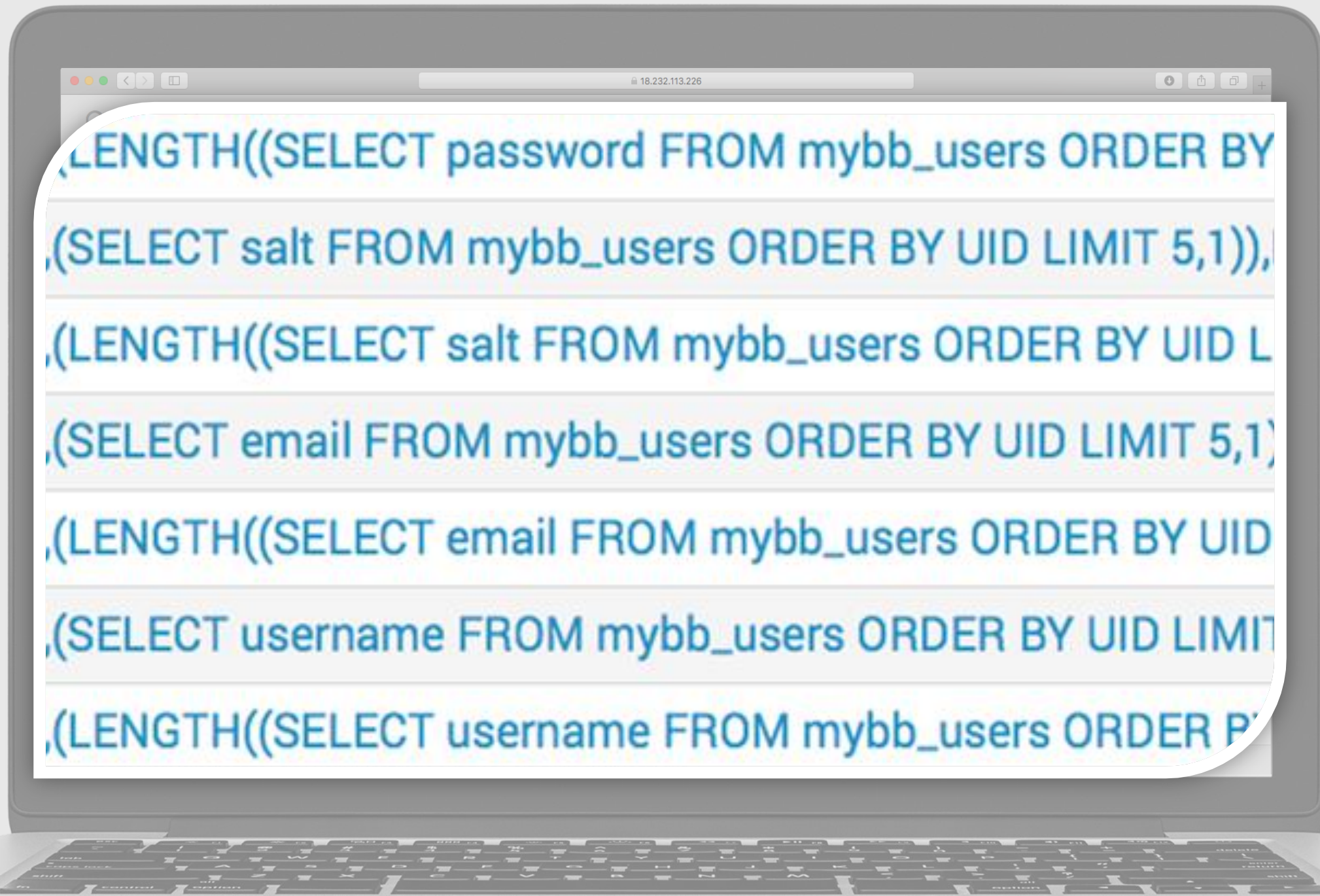
SQL Injection REGEX: Query

- What is the scope?
  - How many attacks occurred?
  - Which were successful?
  - What data was targeted?
  - What data was exposed?

SQL Injection REGEX: Query

For simplicity, pull out the query and result separately.

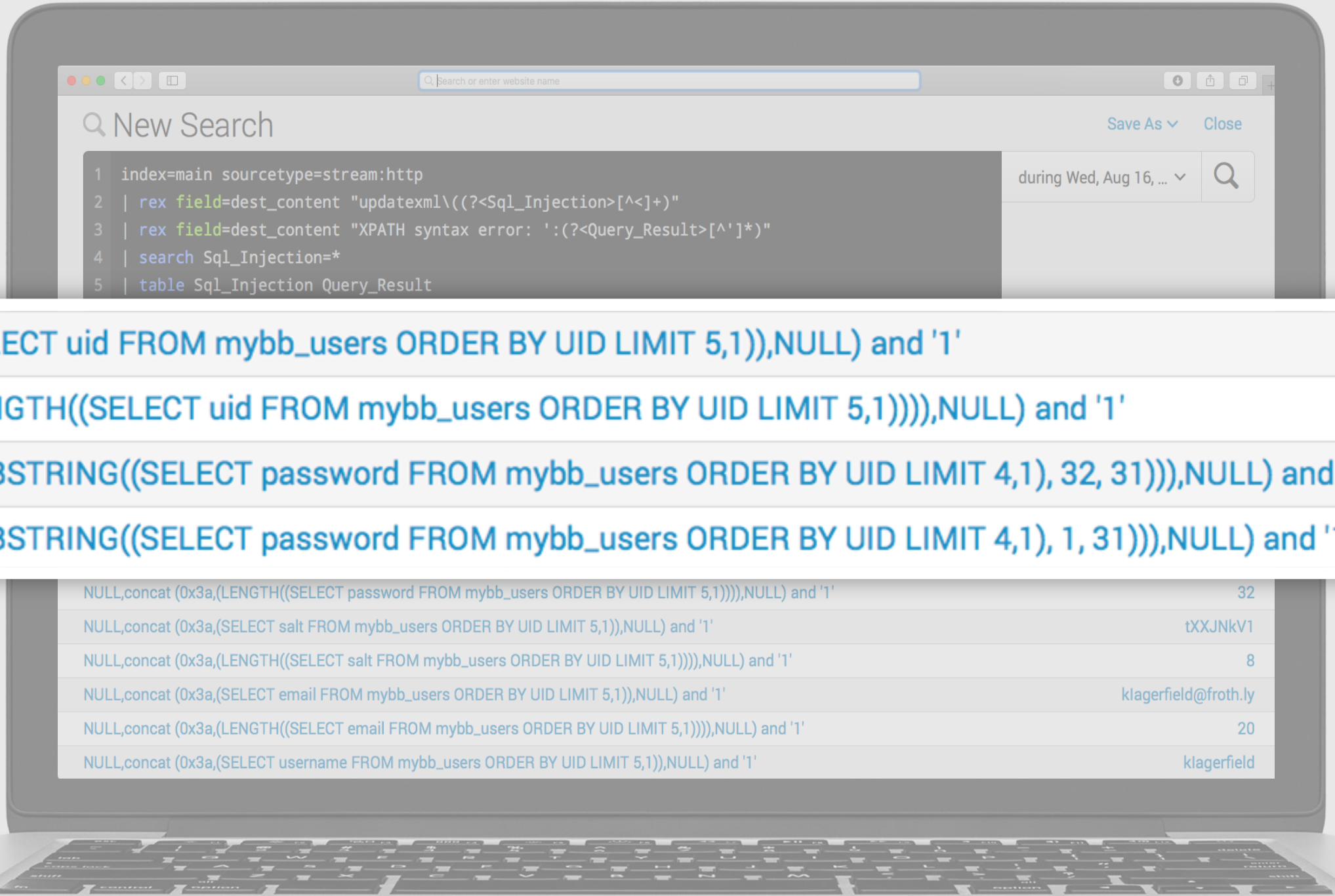
```
index=main sourcetype=stream:http
| rex field=dest_content "updatexml\((?<Sql_Injection>[^<]+)”
| search Sql_Injection=*
| table Sql_Injection
```

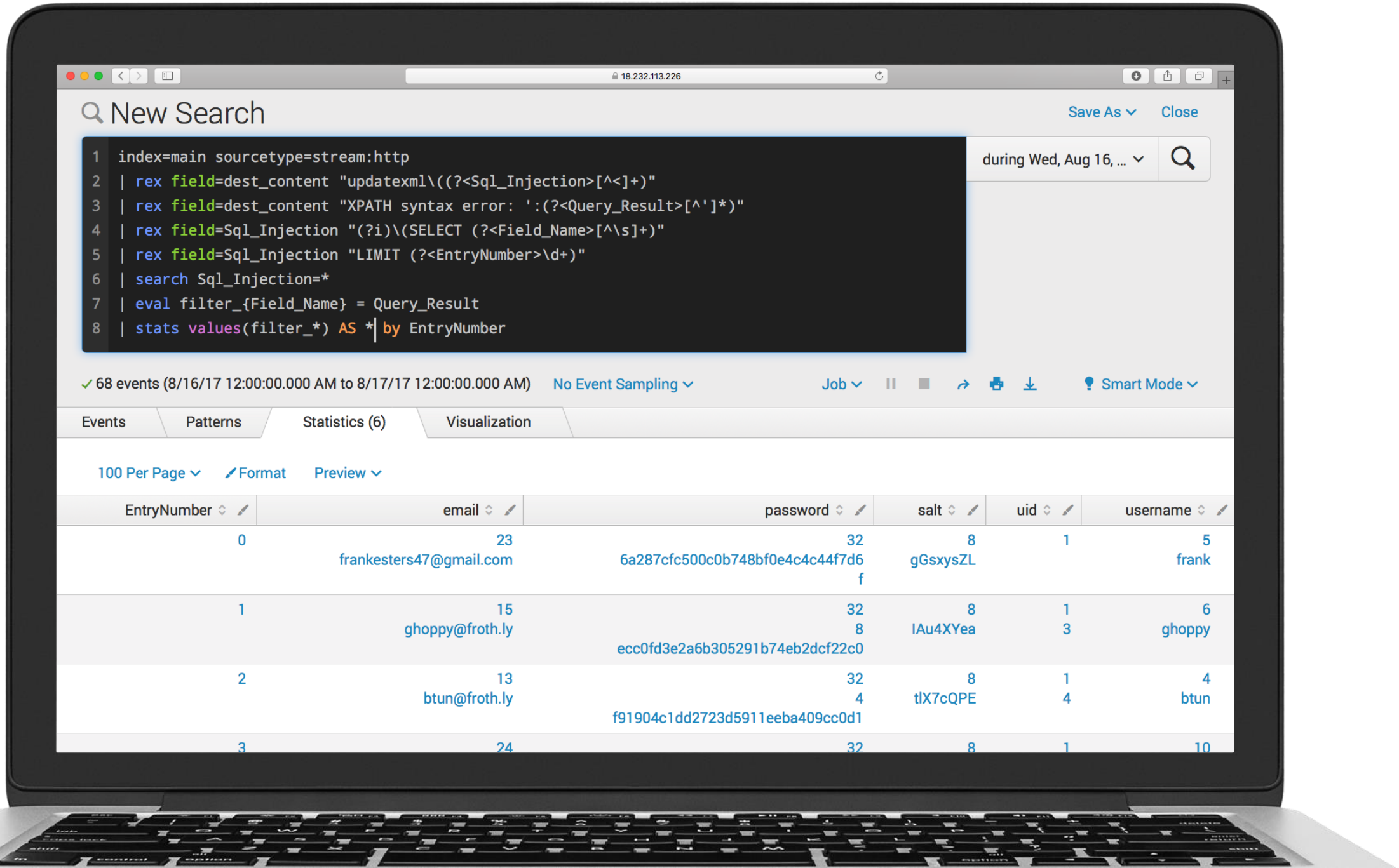


SQL Injection REGEX - Data

Now extract the results.

```
index=main sourcetype=stream:http
| rex field=dest_content
  "updatexml\((?<Sql_Injection>[^<]+)”
| rex field=dest_content "XPATH syntax error: ‘:
  (?<Query_Result>[^']*)"
| search Sql_Injection=*
| table Sql_Injection Query_Result
```

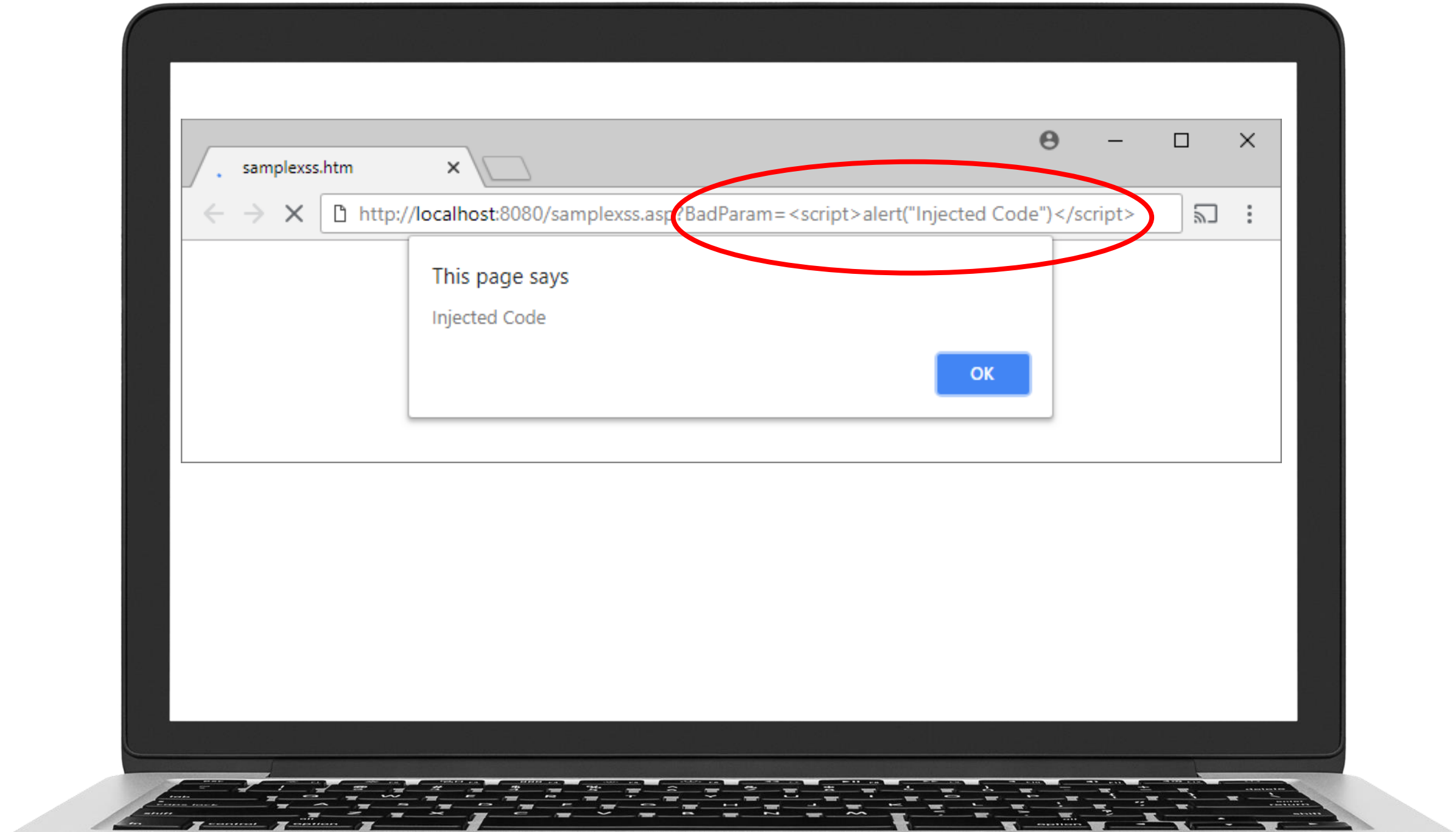


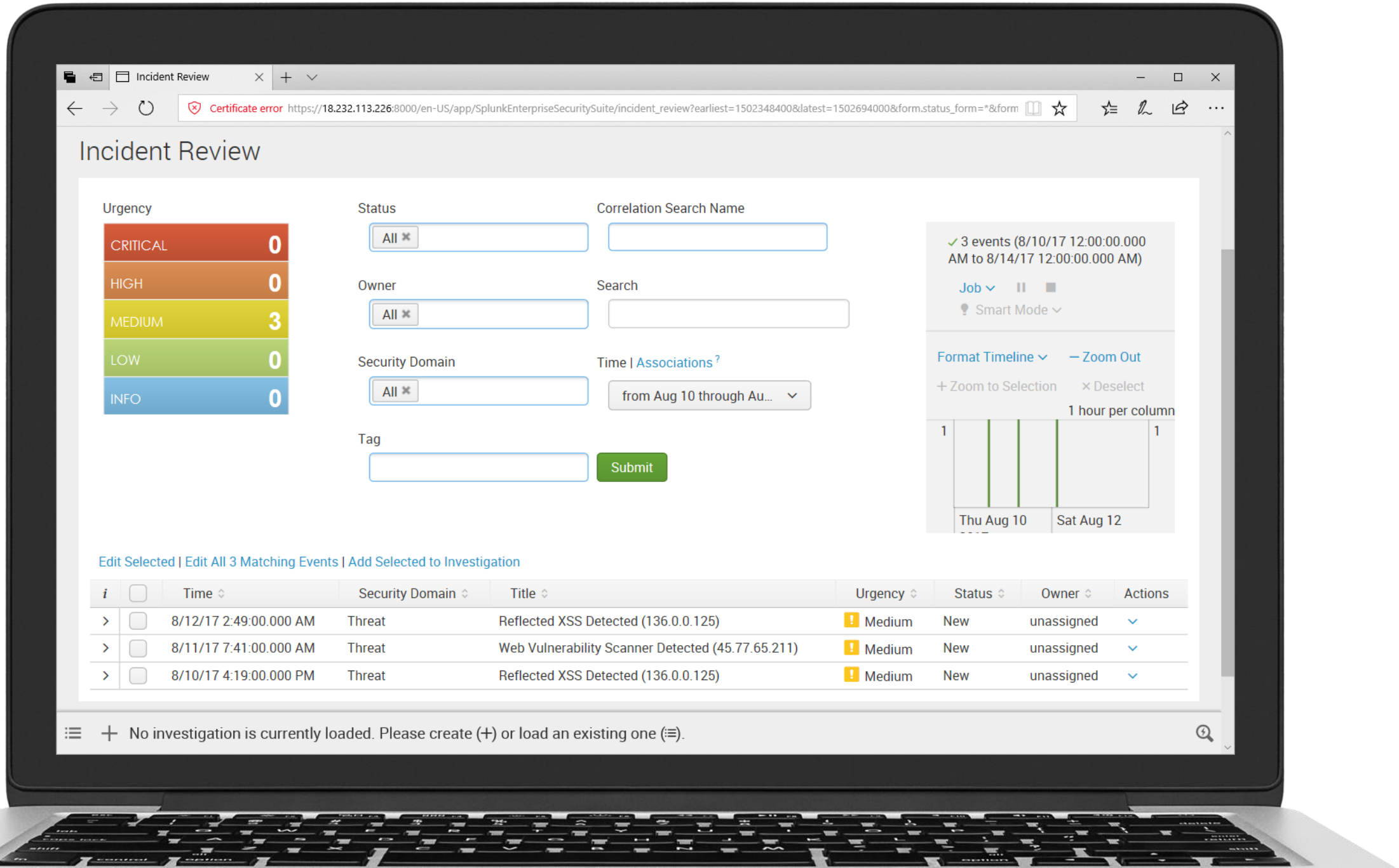


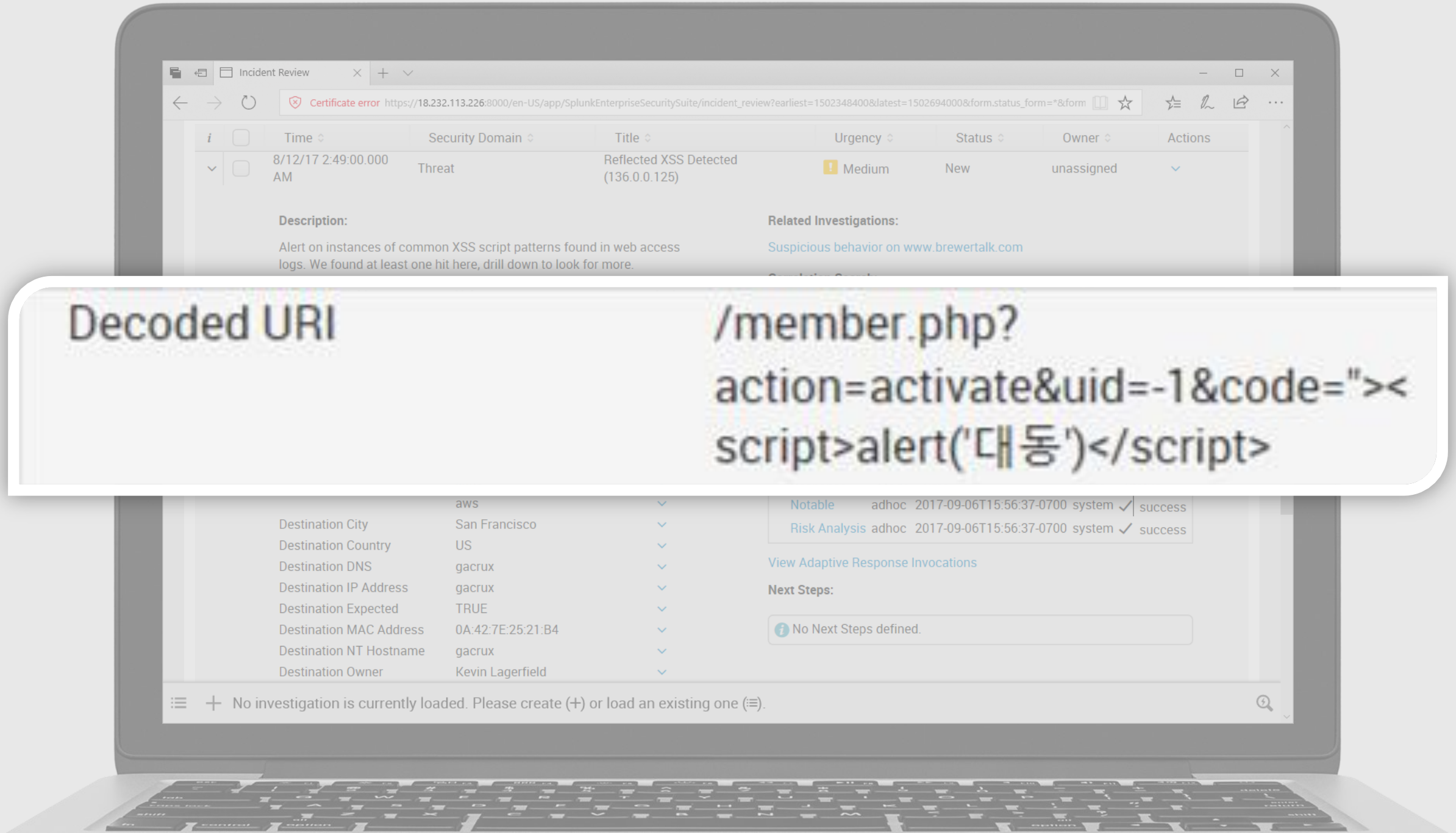
Cross Site Scripting

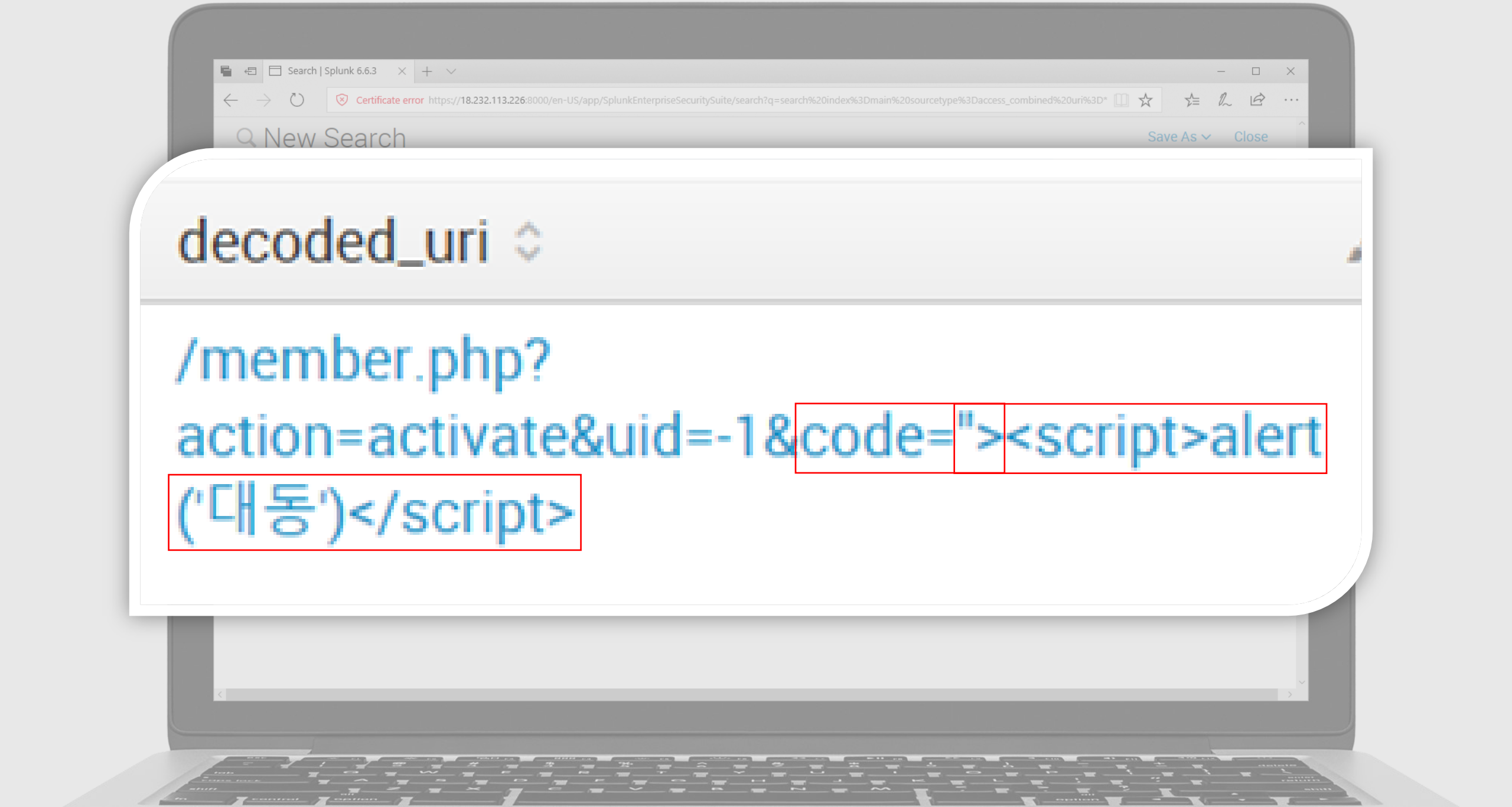
Incident Response

splunk> .conf18





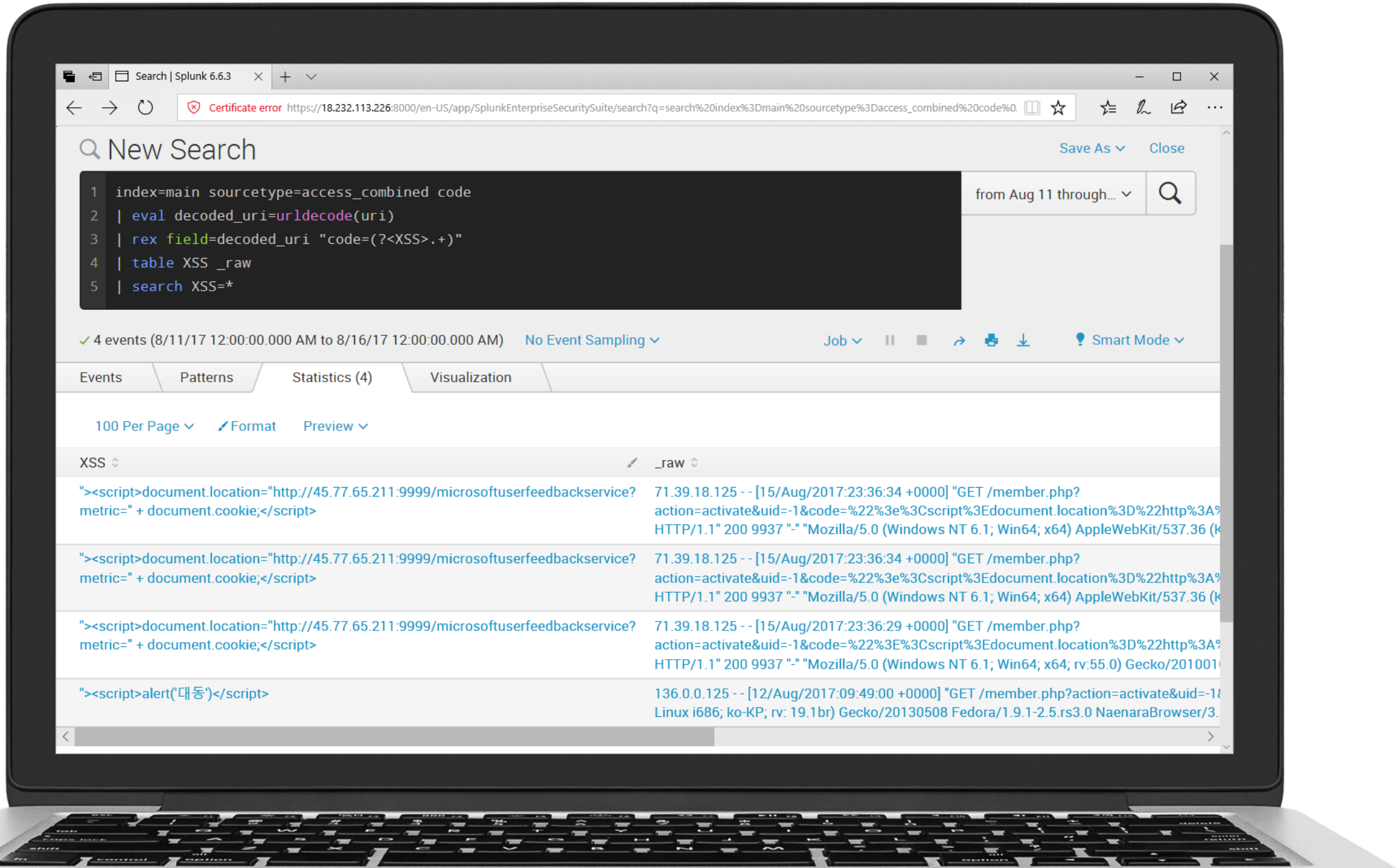




XSS REGEX

- `| rex field=decoded_uri “code=(?<XSS>.+)”`

splunk> .conf18



Which is easier to understand?

```
1  index=main sourcetype=access_combined uri=*code*
2  | eval decoded_uri=urldecode(uri)
3  | search decoded_uri="*code=\">*"
4  | table _time decoded_uri
```

```
1  index=main sourcetype=access_combined code
2  | eval decoded_uri=urldecode(uri)
3  | rex field=decoded_uri "code=(?<XSS>.+)"
4  | table XSS _raw
5  | search XSS=*
```

Working with Sysmon

Incident Response

splunk> .conf18

Sysmon

- According to Microsoft, sysmon "...provides detailed information about process creations, network connections, and changes to file creation time."
- Useful for identifying
  - Downloads
  - Suspicious processes
  - Unusual activity

Scenario

- Internal employee is hiding browsing history using Tor. What is the version of the Tor browser?
- What do we know?
  - Workstation: wrk-aturing
  - Keyword: "tor"

Sysmon

Sample Log:

- <Channel>Microsoft-Windows-Sysmon/Operational</Channel>

  <Computer>wrk-abungst.frothly.local**</Computer>**

  <Security UserID='S-1-5-18'/></System>

  <EventData>

  <Data Name='UtcTime'>2017-08-27 03:57:15.174</Data>

  <Data Name='Image'>C:\Program Files\Splunk\bin\splunk-admon.exe**</Data>**

  </EventData>

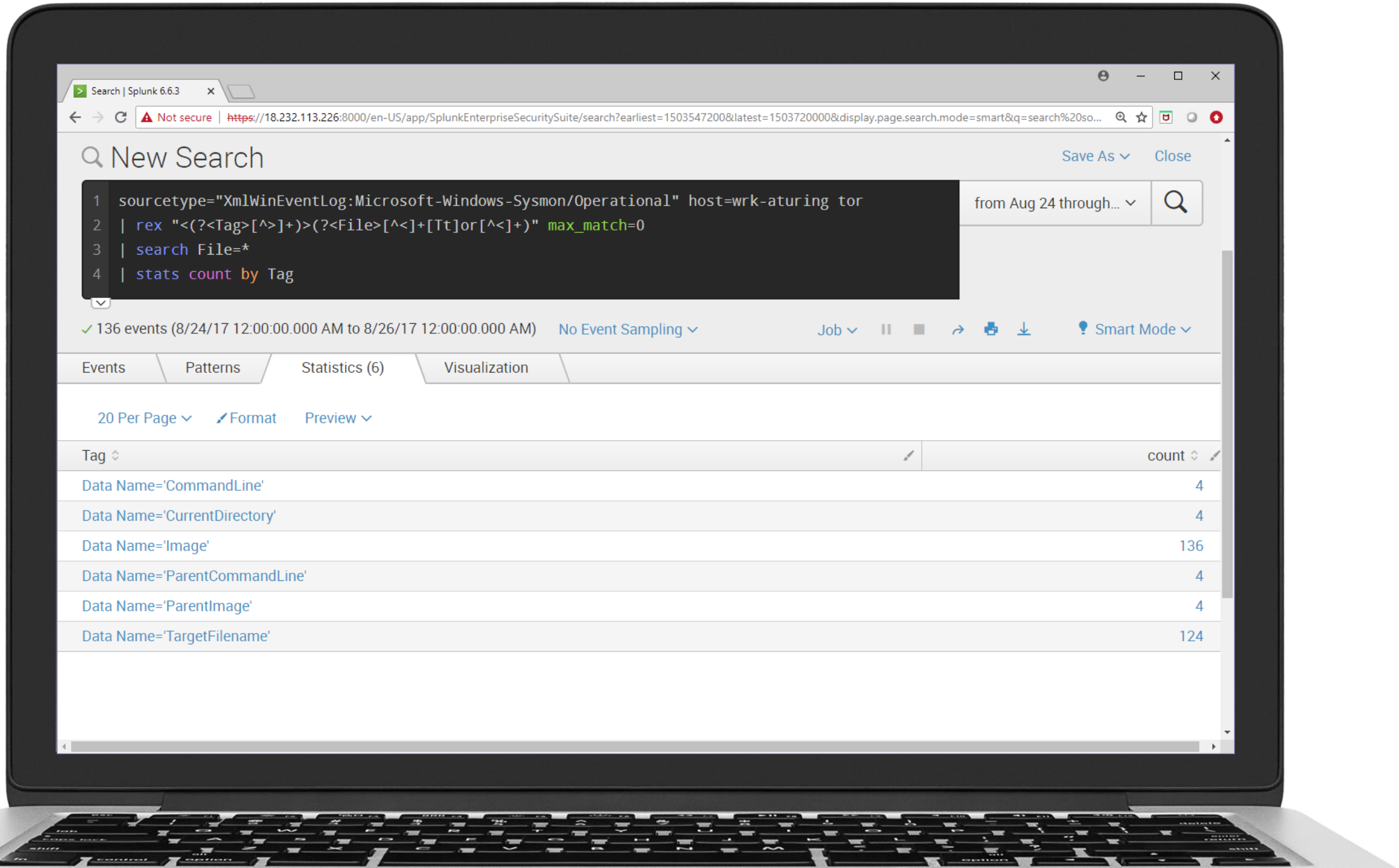
- Each data element has <tag>Data</tag>

Sysmon

Stimulus: Searching for Suspicious Download

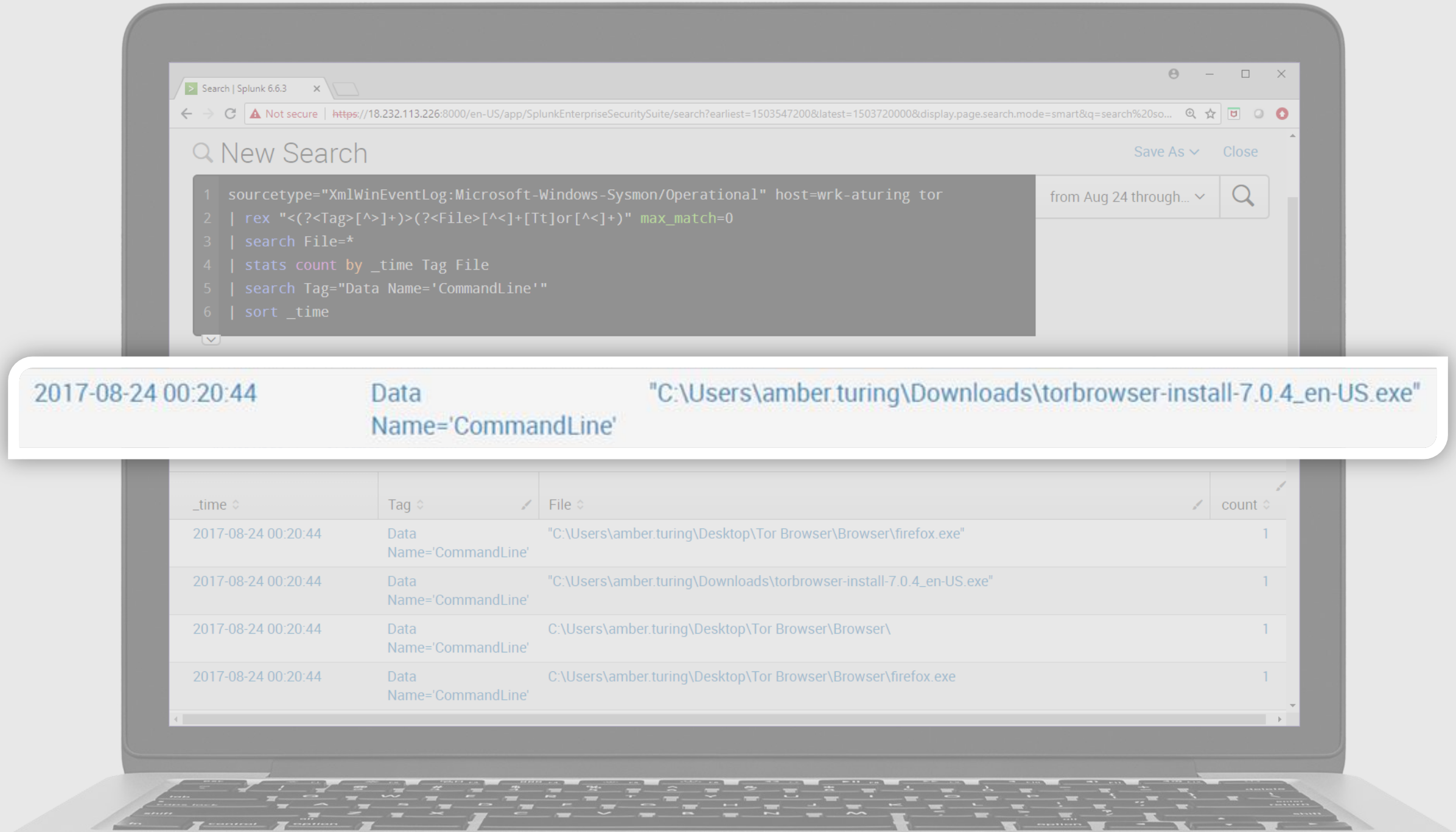
Keyword: Tor

```
sourcetype="XmlWinEventLog:Microsoft-Windows-
  Sysmon/Operational" host=wrk-aturing tor
| rex "<(?<Tag>[^>]+)>(?<File>[^<]+[Tt]or[^<]+)" max_match=0
| search File=*
| stats count by Tag
```



Sysmon

```
sourcetype="XmlWinEventLog:Microsoft-Windows-
  Sysmon/Operational" host=wrk-aturing tor
| rex "<(?<Tag>[^>]+)>(?<File>[^<]+[Tt]or[^<]+)" max_match=0
| search File=*
| stats count by _time Tag File
| search Tag="Data Name='CommandLine'"
| sort _time
```



Q&A

splunk> .conf18

Thank You

Don't forget to rate this session in the .conf18 mobile app