Security Ninjutsu Part Three: Real World Correlation Searches

David Veuve

Staff Security Strategist, Splunk

Disclaimer

During the course of this presentation, we may make forward looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC. The forward-looking statements made in the this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not, be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

Agenda

1. Intro + Content from Prior Years
2. Correlation Search Goals (small shops, big SOCs)
3. Approaches to Building Correlation Searches

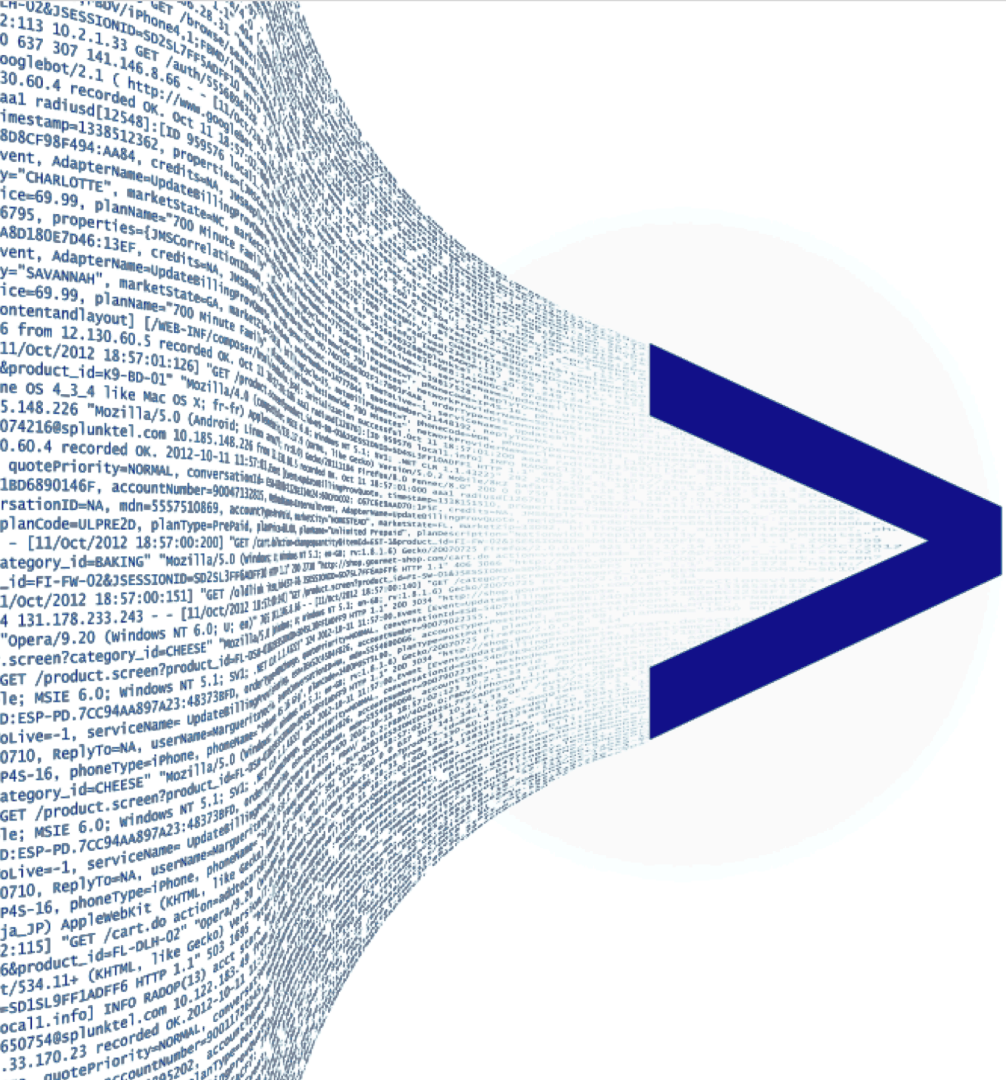
4. Simple Single Scenario Searches
5. Multi Sourcetype Searches
6. Generalized Multi-Tactic Searches
7. Advanced Logic Searches
8. Statistical Behavioral Searches

Personal Introduction

- **David Veuve** – Staff Security Strategist, Security Product Adoption
- SME for Security, Architecture, Analytics
- dveuve@splunk.com
- Former Splunk Customer (For 3 years, 3.x through 4.3)
- Primary author of Search Activity app
- Former Talks:
  - **How to Scale Search from _raw to tstats: .conf 2016 (This year!)**
  - Security Ninjutsu Part Two: .conf 2015
  - Security Ninjutsu Part One: .conf 2014
  - Passwords are for Chumps: .conf 2014

When You Say “Real-World”

- All searches received directly from Splunk Customers
- All SOCs are of medium to high sophistication
- Four different models of SOC
- Six customers contributed in some fashion. Average employee count: 93k (stdev: 103k – we take all kinds)
- Four different industries



Past Security Ninjutsu

The Splunk Portfolio

Splunk Premium Solutions

- Splunk IT Service Intelligence™
- Splunk Enterprise Security™
- Splunk User Behavior Analytics™

Rich Ecosystem of Apps & Add-Ons

Microsoft .net, Symantec, cisco, salesforce.com, Kepware, NetApp, Linux

splunk>enterprise | splunk>cloud | splunk>light | Hunk

splunk> Platform for Operational Intelligence

- Forwarders
- Syslog/TCP
- Mobile
- IoT Devices
- Network Wire Data
- Hadoop
- Relational Databases
- Mainframe Data

Splunk Correlation Rules

- Easy in Enterprise Security
- But even in core Splunk Enterprise, any search can:
  - Send an email
  - Trigger ServiceNow/Remedy/any other ticketing system
  - Run a script
  - Interact with other systems to block / increase logging for hosts
- Correlation in Splunk is just searching

Security Ninjutsu 2014

- Heavy focus on overall lifecycle including visibility, analysis, action
- Mini “how to accelerate search”
- Example Scenarios:
  - Threat Intel
  - Detecting File Hash that Doesn’t Belong
  - Detecting a Dr/Nurse/DBA Viewing Too Many Patient Records
  - Visual Event Correlation

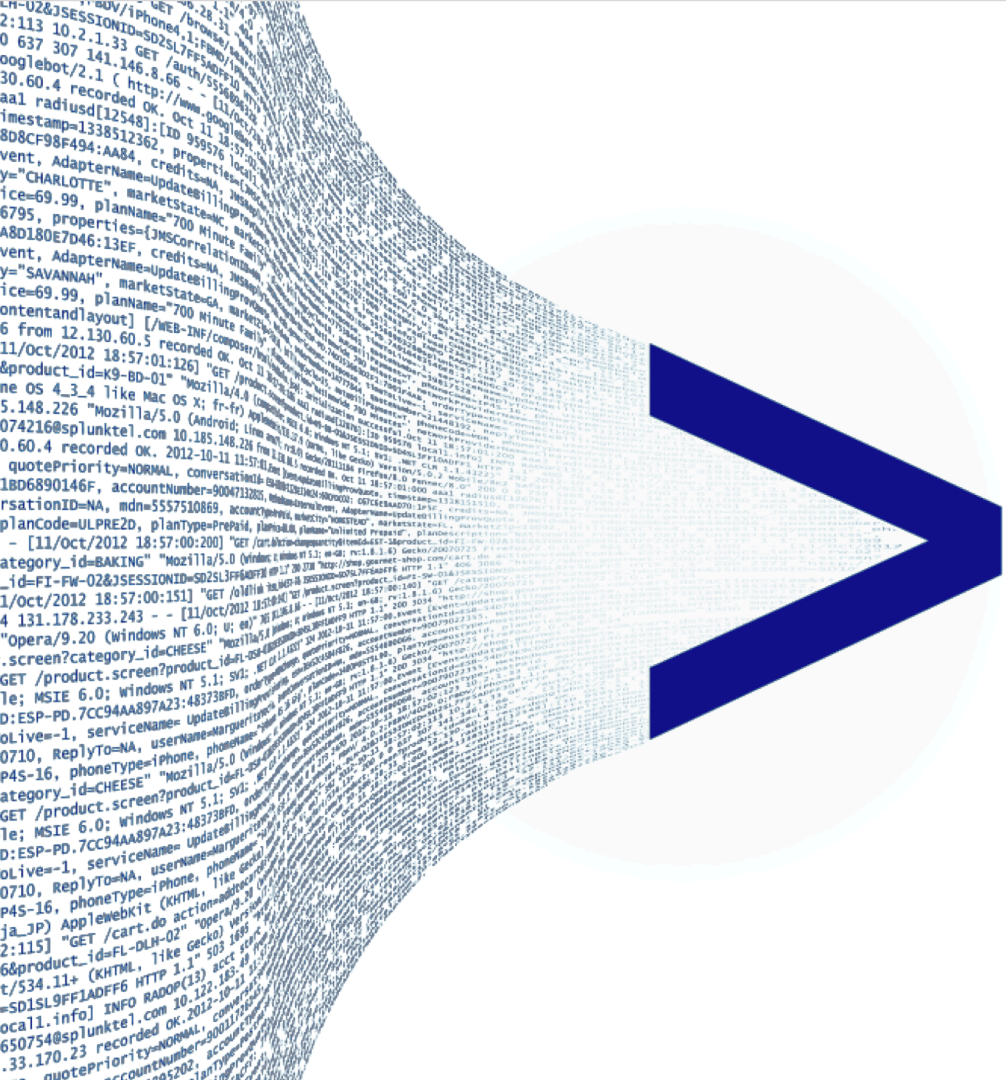
Security Ninjutsu 2015

- Heavy focus on correlation searches
- Focus on three concepts, with a few supporting examples (and less explanation of SPL):
  - Correlation Across Multiple Sourcetypes
  - Risk + Analytics (in the form of Privileged User Monitoring)
  - Conquering Alert Fatigue

Tstats Not Explained Here

- One of my traditional goals: get more people to use tstats
- It will make your searches faster, better, and let you ask more questions
  - Asking Questions is half the value of Splunk

- Explained in ample detail in:
- How to Scale: From _raw to tstats!



Correlation Search Goals

Goal Of A Correlation Search

Actual Goals Of A Correlation Search

- A search should detect bad or suspicious behavior while also providing the necessary context and confidence to balance the urgency of the detected behavior with the availability of the proper response
- Key components that can render a search useless:
  - Insufficient Confidence
  - Insufficient Context
  - Too Many Alerts to Manage
  - Inaccurate Detection
  - Incomplete or Incorrect Urgency

Correlation Search Response Types

- Immediate Action (e.g., ransomware)
- Investigate (e.g., new process never seen before)
- Accrue Indications (e.g., logon anomalies)

Organizational Structures

- There can be many different structures, but showcasing four:
- Mini-SOCs
  - Jane is our analyst, director, CISO, and doesn't sleep
- Traditional Organization (Technology Company)
  - Tier 1 – Tier 2 – Tier 3
- Super-Hunting
  - Teams
- Super-Process
  - Tier 2 + Tier 3

Mini-SOCs

- For smaller customers, with only one or two analysts
- Heavy requirement for high confidence correlation searches
- Tend to be driven more by anti-virus alerts than security threat detection due to time and difficulty
- Can be good a candidate for a technology like UBA to outsource some of that security logic

Traditional Organization

- The most common SOC structure today
- Tier 1 are junior analysts who attempt to do quick validation for false positive identification, or deal with simple issues.
  - Primary goal is to close or escalate inside of 15 minutes.
  - May be outsourced to MSSP for resource-constrained customers
- Tier 2 are “standard” analysts with 1-3 years of experience
  - Deal with escalations from Tier 1
- Tier 3 are advanced analysts with 3-10 years of experience
  - Deal with the highest level incidents, may also do hunting
  - Can be outsourced to parent companies, or security consulting companies

Traditional Organization – CS Impacts

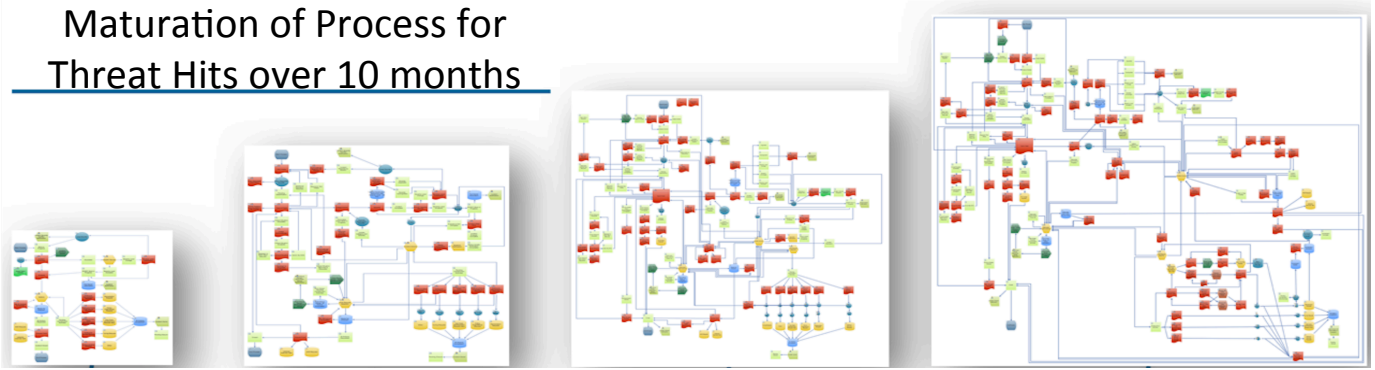
- Basically none – because this is the bulk of SOCs, this is the ideal world for traditional correlation searches

Super-Hunting

- One of our sample customers structures their SOC in hunting PODs
- Each pod has one comparatively junior analysts and one senior
- They have around 200 correlation searches saved, but ~60 running at any point in time
  - This is a pretty typical number for most SOCs
- They tend to focus less on attacks for specific patterns and more on multi-tactic tools, higher level analytics

Super Process

- Another customer is super-process oriented, and has no tier 1
- They hire only tier 2 / tier 3, and leverage that security skill along with heavy process to iterate quickly on their correlation searches

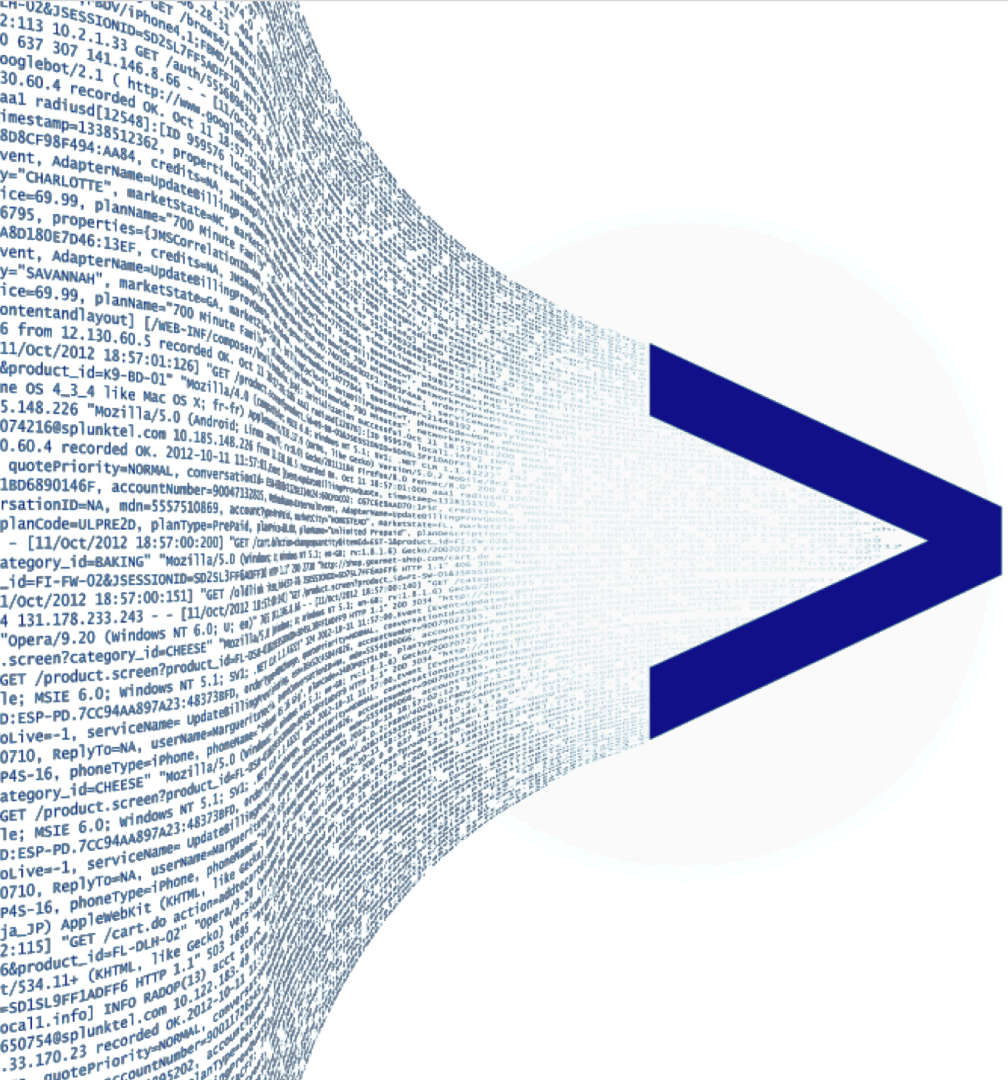


A Good Correlation Search understands Visibility, Analysis, AND Action

Visibility – Analysis – Action

- Framework for evaluating data and responding Splunk
- Applies to all existing frameworks, as it's the Splunk side of the loop
- For example, Let's look at the lateral movement section of the kill chain
  (Not familiar with the kill chain? It's a great way to understand the phases of an attack. Check the URL below.)
- Visibility: What data will let you detect Lateral Movement?
- Analysis: What will you do to that data to come to a decision?
- Action: What will you do in response to that decision?

Kill Chain: http://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf



Approaches to Building Correlation Searches

What Do You Build? Why? How?

- Specific Searches for Specific Scenarios
  - E.g., spam campaign subjects, IOCs
- Specific Searches for General Tactics
  - E.g., tools that attackers tend to use
- High Confidence Behavioral Searches
  - E.g., tracking the geoip ranges for executives and searching for logins outside of them – land speed violations on steroids
- Risk Indicators / High Noise Generation Searches
  - E.g., users logging into new systems, etc.
- Noise Reducing Statistical / Behavioral Searches
  - E.g., ES Risk Framework, UBA, or materials from .conf2015 Security Ninjutsu

Ordered from most common / simplest to least common / most advanced

Specific Scenarios

- These are always a direct response to an actual incident (major or minor)
- Spearphishing, ransomware IOCs, etc
- Most customers have 10-30 of these running at any point in time
- Because these are relatively acute, they typically run frequently over short time periods
- They typically get pushed out with relatively little QA, but have little risk due to simple logic, and can be iterated quickly as needed
- Once the outbreak dies off, these searches are reduced in frequency or killed off
  - E.g., “Symantec now offers protection, we don’t need the search anymore”

Specific Searches For General Scenarios

- These are searches for specific TTPs, Tools Techniques and Procedures (some specific to a threat actor, but usually very general)
- These tools can come out of incidents, but they're as likely to come out of security research
- Development process is longer due to heavier QA to adapt security research to a specific organization
- During dev, alerts will not hit SOC, and by the time the SOC gets access most iteration is complete
- They tend to live longer, particularly the more general ones

High Confidence Analytics Searches

- A great example was from Drew Hunt from Bechtel at .conf2014 was land speed violations (superman problem) on steroids:
  http://conf.splunk.com/session/2014/conf2014_AndrewHunt_Bechtel_Security.pdf
  http://conf.splunk.com/session/2014/conf2014_AndrewHunt_Bechtel_Security.mp4
- These have a very heavy development cycle, with very long QA periods before the alerts go to the SOC
- Typically, one or two people from the SOC will be involved in the later stages of the QA process to validate the alert is actionable, given that it may be less easily understood
- Iteration and usage may never end for these searches depending on the scenario

Risk Indicators/High Noise Searches

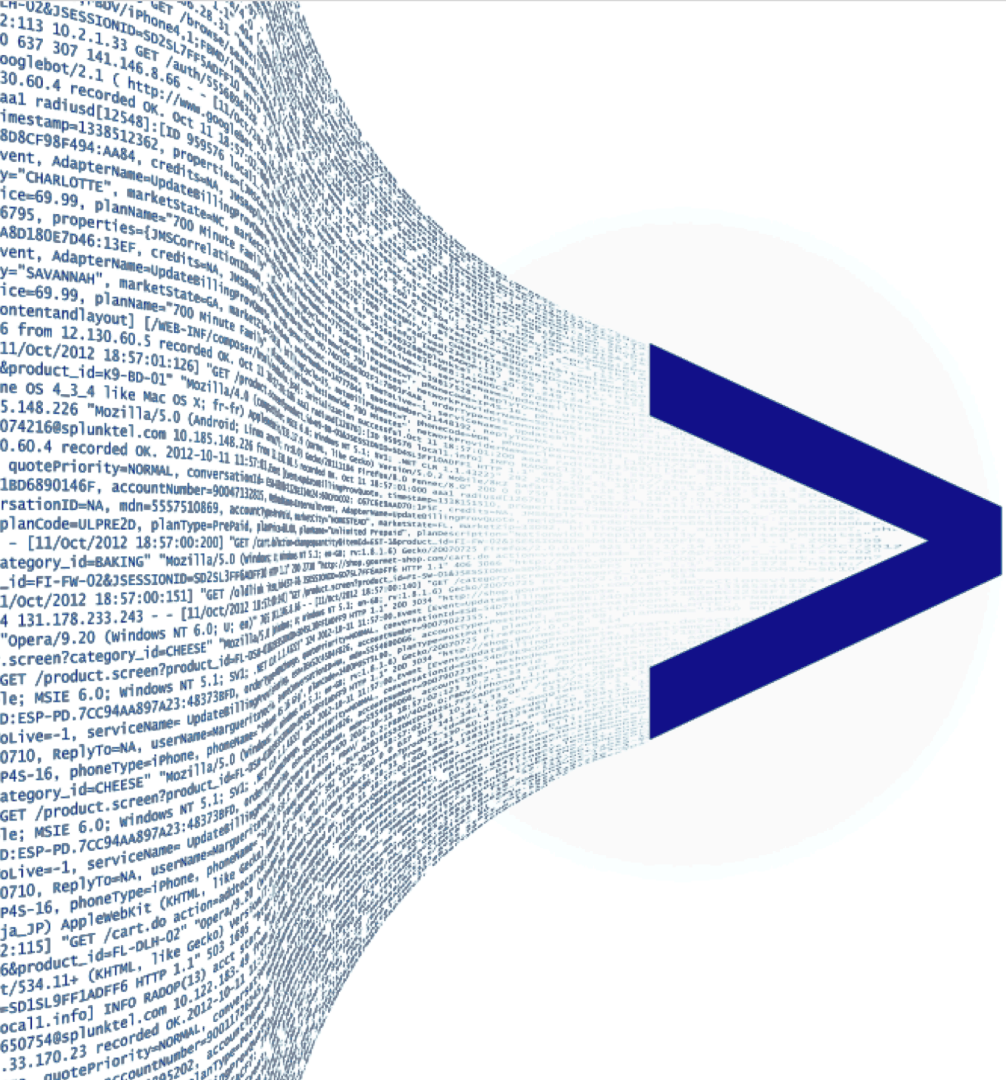
- These can be generated by searches (such as when a user logs into a new system) or generated by security tools (such as IDS alerts)
- Results will be so noisy to make them unusable by the SOC directly
- That said, they're extremely valuable for detecting advanced attackers / new TTPs if fed into a statistical / ML / behavioral search
- These searches are less common because few organizations are able to use them
- Because the goal is not to reduce false positives, iterations are few
- Driven mostly by security research (rightly or wrongly)

Noise Reducing Statistical/Behavioral Searches

- If you have hundreds or thousands of alerts to review, your SOC can't keep up. SOCs need to analyze for anomalous alerts
- Approach (1): Generic anomaly detection algorithms
- Approach (2): ES Risk framework (naïve analytics tied to entities)
- Approach (3): Splunk UBA Threat Models (machine learning!)
- Approach (4): Statistical searches (build your own alert logic)
- For more on this, visit .conf2015 Security Ninjutsu Part Two:
  http://conf.splunk.com/session/2015/recordings/2015-splunk-130b.mp4
  http://conf.splunk.com/session/2015/conf2015_DVeuve_Splunk_SecurityCompliance_SecurityJiujitsuBuildingCorrelation.pdf

On Creating Advanced Searches

- Anyone using Splunk can create the searches for Specific Scenarios. The only expertise is around scheduling searches (how many, how often)
- Advanced searches require someone with an analytical mindset, frequently with some casual scripting / development in their past
  - David's Opinion: Don't call the people who build these searches "data scientists" – they're data nerds at most! Splunk doesn't usually require actual data scientists, and when we toss around that term we make people think they can't do it. Avg+Stdev+Latest isn't hard. Go see my tstats talk



Specific Scenario

Ransomware - Requirements

- Many correlation searches in Splunk are very simple
- In this case, it was the real world application of research to detect ransomware leveraging threat intelligence available on Google (file extensions are pretty easy to find)
- We are looking for any log messages referencing ransomware so that those can be escalated and dealt with aggressively as a high priority ticket

Ransomware – Visibility

- There are two general sources of visibility for this data – Windows audit logs, and endpoint security products
  - Windows Audit Logs can be configured to track file writes. This is not uncommon on file shares to deal with HR / Helpdesk type needs, but rare on workstations due to the event volume
  - Endpoint security products can also be very useful here, particularly Host-based Intrusion Detection Systems where they won't necessarily block everything but they can log many suspicious things
- For this customer, they used Symantec HIDS as the primary eventing source

Ransomware - Action

- When this triggers, SOC looks for Windows 4688 (Process Launch) logs to see what was launched around the same time
- Engage Live Response: fire up grr or encase to see where the files are being created, what the file names are
  - Goal is to figure out what ransomware it is (Locky, etc.)
  - Use Sentinel, or psexec to the box to kill the locker itself
  - Look for signs of data exfil in FW logs
    - Oftentimes they can beat the exfil timer because the exfil is occurring on encrypted data to prevent you from pulling out of a pcap
- Analyze the malware itself
  - Send to file analysis framework (for this customer, irma, but also consider stoq and it's Splunk app by @meansec or rkovar@splunk.com!)
  - Look for C2, Domains, any signs of infrastructure
  - Look for traffic going to those IOCs company-wide
- Reimage User Laptop

Ransomware – High Level Search

- index=sep* (sourcetype="sep12:risk" OR sourcetype="sep12:proactive")
  - Pull the dataset that provides visibility
- NOT (signature="EICAR test file")
  - Exclude any false positives – the only one seen was with test cases.
- (.ecc OR .ezz OR .exx OR .zzz OR .xyz OR .aaa OR .abc OR .ccc OR .vvv OR .xxx OR .ttt OR .micro OR .encrypted OR .locked OR .crypto OR _crypt OR .crinf OR .r5a OR .XRNT OR .XTBL OR .crypt OR .R16M01D05 OR .pzdc OR .good OR .LOL! OR .OMG! OR .RDM OR .RRK [….. MANY items truncated ……])
  - Filter for the file IOCs
- | table _time, *_nt_host, dest_ip, user, signature, file_size, occurrences, process, actual_action, secondary_action, confidence, hash_value
  - Prepare the relevant indicators

Ransomware – Actual Search

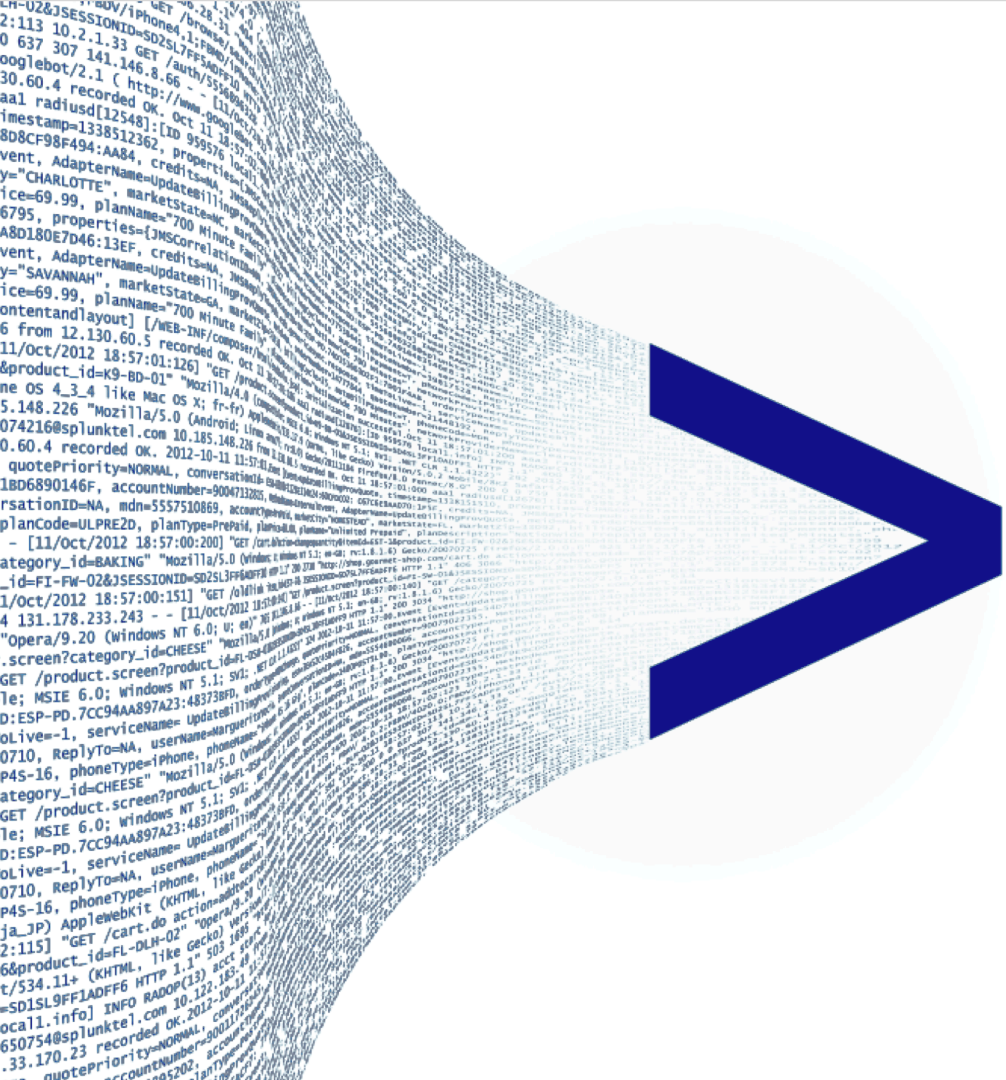
- index=sep*
- (sourcetype="sep12:risk" OR sourcetype="sep12:proactive")
- NOT (signature="EICAR test file")
- (.ecc OR .ezz OR .exx OR .zzz OR .xyz OR .aaa OR .abc OR .ccc OR .vvv OR .xxx OR .ttt OR .micro OR .encrypted OR .locked OR .crypto OR _crypt OR .crinf OR .r5a OR .XRNT OR .XTBL OR .crypt OR .R16M01D05 OR .pzdc OR .good OR .LOL! OR .OMG! OR .RDM OR .RRK OR .encryptedRSA OR .crjoker OR .EnCiPhErEd OR .LeChiffre OR .keybtc@inbox_com OR .0x0 OR .bleep OR .1999 OR .vault OR .HA3 OR .toxcrypt OR .magic OR .SUPERCRYPT OR .CTBL OR .CTB2 OR .locky OR "*ReCoVeRy*" OR HELPDECRYPT.TXT OR HELP_YOUR_FILES.TXT OR HELP_TO_DECRYPT_YOUR_FILES.txt OR RECOVERY_KEY.txt OR HELP_RESTORE_FILES.txt OR HELP_RECOVER_FILES.txt OR HELP_TO_SAVE_FILES.txt OR [….. TRUNCATED ……])
- | table event_time, _time, src_nt_host, dest_nt_host, dest_ip, user, signature, file_size, occurrences, process, actual_action, secondary_action, confidence, hash_value

Ransomware – Next Level

- For analyzing malware (and also for when people receive phishing emails) integrate open source file analysis frameworks into your Splunk ecosystem for epic results and almost no license
- This customer used IRMA and modified it heavily – likely easier to use the work for Ryan Kovar and Marcus LaFerrera via stoQ:
- https://files.sans.org/summit/Digital_Forensics_and_Incident_Response_Summit_2016/PDFs/stoQing-Your-Splunk-Ryan-Kovar-and-Marcus-LaFerrera.pdf
- https://splunkbase.splunk.com/app/3196/

Ransomware @ .conf

- James Brodsky + Dimitri McKay, Splunking The Endpoint: Wed @ 3:30, Thurs @ 12:25
- Ken Westin, Ransomware Wrangling with Splunk: Next!
- Palo Alto Networks, PAN and Splunk Together to Prevent Attacks and Protect Your Data: Wed @ 11
- Boss of the SOC (hint, ransomware exists!)



General Scenarios

General Searches

- Rather than looking for one specific IOC or toolset (e.g., “let’s add the file extensions used by the latest variety of locky”), looks for techniques used by attackers like clearing the event log
- More likely to be inspired by research than incidents
- Write one search to monitor a category of risky events
- Typically would send events to SOC
  - For higher noise, could also contribute to risk
  - For both low and high noise, can also send directly to UBA

The svchost That Wasn’t - Backstory

- Tier 3 analyst was teaching a tier 1 analyst how the windows process tree works – system proc is PID 4, services.exe spawns svchost.exe, which owns other services, etc.
- In showing on live data, found a process that was called ”svchost.exe” but wasn’t spawned off services.exe

The svchost That Wasn't - Requirements

- Initially they looked for svchost.exe that wasn't owned by services.exe
- After testing, they found rare but legit exceptions (akin to a non-root daemon on linux spawning a root process)
- Now they look for svchost.exe owned by non-services.exe and filter results to exclude the md5 for versions of svchost.exe that are actually signed by MS
  - Pulled this list based on research in their Splunk and validation with the Windows File Signing check in Explorer

The svchost That Wasn't - Visibility

- There are two different approaches that I have seen for pulling this process launch data.
- By far, the most common is to use Microsoft sysmon (you can filter to include only events you care about)
  - Want to know how this works? Go to:
    James Brodsky + Dimitri McKay, Splunking The Endpoint: Wed @ 3:30, Thurs @ 12:25
- This particular customer is using Windows Event ID 4688 which logs process paths, and uses 3rd party Windows Logging Service for context

Sysmon will log:

- User
- Process Name, CLI Args, PID, GUID
- Parent Process Name, CLI Args, PID, GUID
- Working Directory
- Time
- Host
- MD5
- Sha1
- Sha256
- IMPHASH

The svchost That Wasn't – Action

- When this occurs, it will typically be commodity malware
- The SOC starts typical live response procedures.
  - grr (Google Rapid Response) or encase the host to get context
  - See processes that were launched around the same time
  - Etc.

The svchost That Wasn't – High Level Search

- sourcetype=Win*Security EventID=4688 BaseFileName="svchost.exe" NOT CreatorProcessName="services"
  - Include WinSecurity Process Launch logs where the process is svchost.exe and the parent is not "services"
- NOT (MD5="54A47F6B5E09A77E61649109C6A08866" OR […])
  - Exclude known valid md5s
- | sort 0 -_time
  - Make sure the events are sorted correctly
- | table _time, Computer, SubjectDomainName, SubjectUserName, BaseFileName, CommandLine, CompanyName, CreatorProcessName, NewProcessName, FileDescription, FileVersion, MD5
  - Put the relevant fields in the right order for the SOC to digest

The svchost That Wasn't – Actual Search

```
index=wineventlog EventID=4688 BaseFileName="svchost.exe" NOT CreatorProcessName="services"
NOT (MD5="54A47F6B5E09A77E61649109C6A08866" OR
    MD5="A412DEDAC6A1FF7BA06FEB3B6725495E" OR
    MD5="C78655BC80301D76ED4FEF1C1EA40A7D" OR
    MD5="8497852ED44AFF902D502015792D315D" OR
    MD5="A1AEAFC58DF7803B8AA2B09EA93C722F" OR
    MD5="E4CA434F251681590D0538BC21C32D2F" OR
    MD5="E3A2AD05E24105B35E986CF9CB38EC47")
| sort 0 -_time
| table _time, Computer, SubjectDomainName, SubjectUserName, BaseFileName, CommandLine, CompanyName,
CreatorProcessName, NewProcessName, FileDescription, FileVersion, MD5
```

Svchost That Wasn't - Action

- Typical Live Response scenario
- Pull forensic details about what the process is, what files it access, what network destinations it communicates with
- Use grr or encase or others to pull those details

Task Scheduler – Backstory

- This is a classic Windows privilege escalation attack, that affected the customer a few years ago
  - Use Case Developer was reading through a list of the top ten tools used by Windows attackers, saw at.exe and thought, “that sounds familiar”
- at.exe allows a normal user to schedule a task that runs with privileged access
- Deprecated as of Windows 7, but still available until Windows 10
- Based on analyst performing research for common exploitation vectors, and reviewing their past company experiences (“the breach”)

Task Scheduler – Action

- When detected, SOC uses grr (Google Rapid Response) to pull forensic details to see more information about:
  - What process did the scheduling
  - What process is being scheduled
  - What the process does.
- Could also use encase or other tools, depending on the host and the analyst preference
- What does it catch:
  - Heavily focused on sys admins doing dumb things, sec ops staff violating rules

Task Scheduler – Search

- EventID=4688 BaseFileName="at.exe" CommandLine="*"
  - Bring in Process Launch Logs
- NOT CommandLine="*FalsePositiveInducingProgram.exe" NOT CommandLine="at"
  - Exclude one program that created false positives, and ignore users who just type "at" at the CLI to see the help page
- | table _time, Computer, SubjectDomainName, SubjectUserName, BaseFileName, CommandLine, CompanyName, CreatorProcessName, NewProcessName, FileDescription, FileVersion, MD5
  - Print the fields the SOC needs to handle events

You Can rundll32 HTML And Javascript???

- index=windows EventID=4688 rundll32.exe (BaseFileName="rundll32.exe" OR BaseFileName="cmd.exe") (javascript OR RunHTMLApplication)
- | sort 0 -_time
- | table _time, Computer, SubjectDomainName, SubjectUserName, BaseFileName, CommandLine, CompanyName, CreatorProcessName, NewProcessName, FileDescription, FileVersion, MD5

- https://thisissecurity.net/2014/08/20/poweliks-command-line-confusion/
- Consider sysmon as well to detect this
- rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";alert('foo'); Quote from the SOC Analyst who shared this: "^^ WHAT THE #@$&?!"

AccountReport.pdf.xls.exe – Actual Search

- (index=wineventlogs EventID=4688) OR (index=sysmon EventCode=1) BaseFileName="*.doc*.exe" OR BaseFileName="*.xls*.exe" OR BaseFileName="*.pdf.exe"
- NOT (…AllowableUsage…) NOT CompanyName="Adobe Systems, Inc."

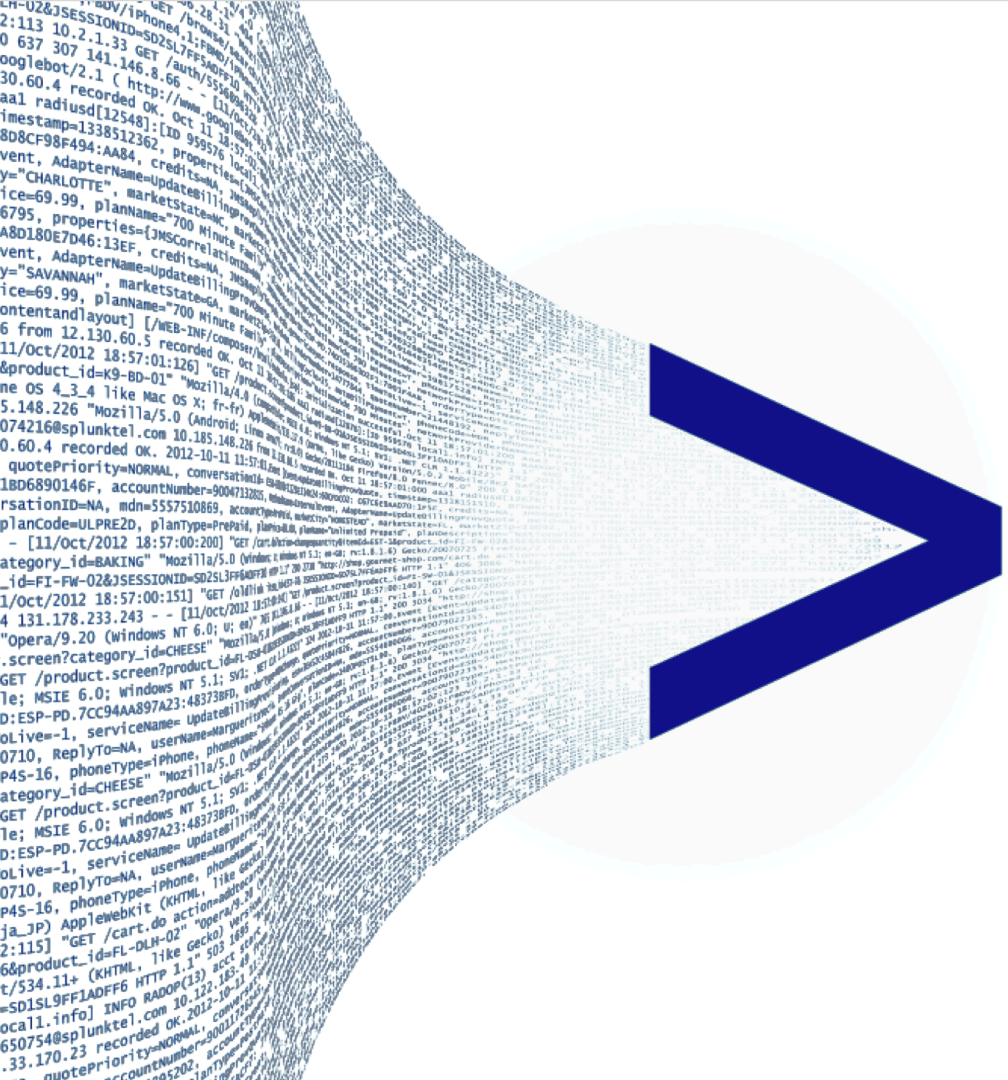
General Scenarios– Next Level

- When choosing what correlation searches of this type to use, you must always make a value / noise tradeoff
- These searches are very low noise and so they go straight to the SOC
- There are many examples that are higher noise – take these and either send them to:
  - ES Risk Framework (or build your own noise reduction search, covered in .conf2015 Security Ninjutsu Part Two)
  - Splunk User Behavior Analytics

When It Doesn't Work – Clearing Event Log

- Clearing the event log is a common hacker technique, and super easy to detect – event id 1102 and 517
- sourcetype=WinEventLog:Security EventCode=517 OR EventCode=1102
- Two customers I asked about this search – neither actually use it
- Too noisy – 50 alerts/week for one customer, 20/week for another

- Look here for meta-analysis, statistical reduction of alert volume
  - Splunk UBA!
  - Also… Security Ninjutsu Part Two from .conf 2014



Detecting Phishing with Splunk

You Can't Phish Me! - Requirements

- Customer wanted to detect phishing attempts
- Two primary risk vectors to consider upfront:
  - Phishers registering similar domain names (e.g., c0mpany.com)
  - Phishers fabricating emails as (it|cio|helpdesk)@company.com
    - Few security controls for this. The few work poorly for a 11,500+ headcount global organization with many outsourced services
- Three correlation searches to address
  - Augment with security tools that identify SMTP Threats

You Can't Phish Me – Search Descriptions

- Search One: Find Similar Domains (c0mpany.com)
  - Use the Levenshtein algorithm packaged in url toolbox app
  - Calculates distance between two strings (company.com and c0mpany.com)
  - Alert for a distance 0 < x < 3 (so company.com and c0mpamy.com)
  - Little tuning required
- Search Two: Find External Emails from company.com
  - Simple detection that looks for incoming emails with sender=company.com
  - Key here is tuning via an inputlookup
- Search Three: Find Internal Emails where from != reply-to
  - Finds internal users fabricating the reply-to

You Can't Phish Me! – Backstory

- Use Case Developer keeps 40-50 searches in test mode at a time
  - Only alerts (him|her) and not the SOC. Validate tuning
- These searches focus on targeted high value phishing (APT-style) but also common ransomware phishing
- Complemented by very heavy Security Awareness Training
  - Periodic Talks with Giveaways
  - Regular Phishing Tests (vendors like phish.me)
  - Major events for Security Awareness Month (October)
  - "When I notify users that they should change their passwords after a third party breach, they forward those emails to IT Security to verify"

You Can't Phish Me! – Visibility

- Phishing detection tends to be relatively straightforward – you want your email logs
- Frequently there are two or more sources of visibility -- internal and external
  - External: Cisco Ironport is the most common that we see among security customers, though there are also many others (including SAAS)
  - Internal: Exchange is the most common that we see, though there are also a few others
- This customer leverages both Ironports and Exchange

You Can't Phish Me! – Actions

- This set of rules trigger about 10-15 HIGH criticality events per day for a 8,000 person company
- SOC immediately analyzes email, who sent it, who else received it, whether that same MTA / domain sent any other messages
- SOC scrubs the email out of the user's inbox quickly to limit impact
- SOC reaches out to end user workstation (live response) to evaluate impact – did the user open, what occurred
- Because this rule runs frequently, the SOC oftentimes will uses these rules to respond to ransomware before the user is affected

You Can’t Phish Me! – Search One High Level

- index=ironport_logs
  - Ingest Ironport logs
- | stats values(*) as * by MID
  - Get a single record per email (Ironport will log many messages)
- | search sender!=company.com
  - Filter to only emails not from our company
- | eval list=".com", ourdomain="company.com"
  - Define our company name, and a list for ut_parse
- |`ut_parse(sender,list)` |`ut_levenshtein(ourdomain, ut_domain)`
  - Use URL Toolbox to parse out the domain, and then compare that to company.com
- | where (ut_levenshtein>0 AND ut_levenshtein<3)
  - Filter for sender domains that are similar but not the same as our domain

You Can’t Phish Me! - Search One Actual Search

- index=messaging
- | fillnull value="-"
- | eval attachment=file_name+"."+attachment_type
- |stats values(sender) as sender values(recipient) as recipient values(message_subject) as message_subject values(attachment) AS attachment values(action) as action by MID
- | search sender!="*@company.com"
- |eval list=".com" |`ut_parse(sender,list)`
- | eval ourdomain="company.com" |`ut_levenshtein(ourdomain, ut_domain)` | where (ut_levenshtein>0 AND ut_levenshtein<3)
- | stats count by sender ut_domain recipient message_subject action ut_levenshtein attachment

You Can't Phish Me! – Search Two High Level

- index=ironport_logs
  - Pull our Ironport Logs
- | transaction MID keepevicted=true
  - Use Transaction to group based on the message ID. (Why not stats?)
- | search "received" sender="*@company.com"
  - Filter to received emails
- | stats count list(recipient) as recipients by received_ip sender message_subject MID _time
  - Summarize the results
- | search NOT [inputlookup lookup.csv | fields received_ip sender]
  - Filter out known exceptions. Lookup.csv will include a valid ip address and sender email address. This returns as follows, to strip out known false positives:
    ((received_ip=8.8.8.8 sender=hr@company.com) OR (received_ip=8.4.4.4 sender=*@company.com))

You Can’t Phish Me! – Search Two Actual Search

- index=messaging
- | transaction MID keepevicted=true
  | search "received"
- | search sender="*@companyname"
- | rex max_match=0 "(?ms)\\[(?<received_ip>\\d+\\.\\d+\\.\\d+\\.\\d+)\\]"
- | stats count list(recipient) as recipients by received_ip sender message_subject MID _time
- | search NOT [inputlookup lookup.csv | fields received_ip sender]
- | sort - count
- | table _time MID received_ip recipients sender message_subject count

You Can't Phish Me! – Search Three High Level

- index=msexchange
  - Unlike the Ironport search, Exchange logs come from inside the network
- | stats list(_time) AS Date list(sender) AS PurportedSender list(resolved_sender) AS ResolvedSender by MID
  - Summarize the email logs into a single event per message ID
- | search PurportedSender=*@company.com PurportedSender!=ResolvedSender ResolvedSender!="<>"
  - Look where it claims to come from company.com, and check to see if that matches the reply-to address.
  - The last filter strips out false positives.

You Can’t Phish Me! – Search Three Actual Search

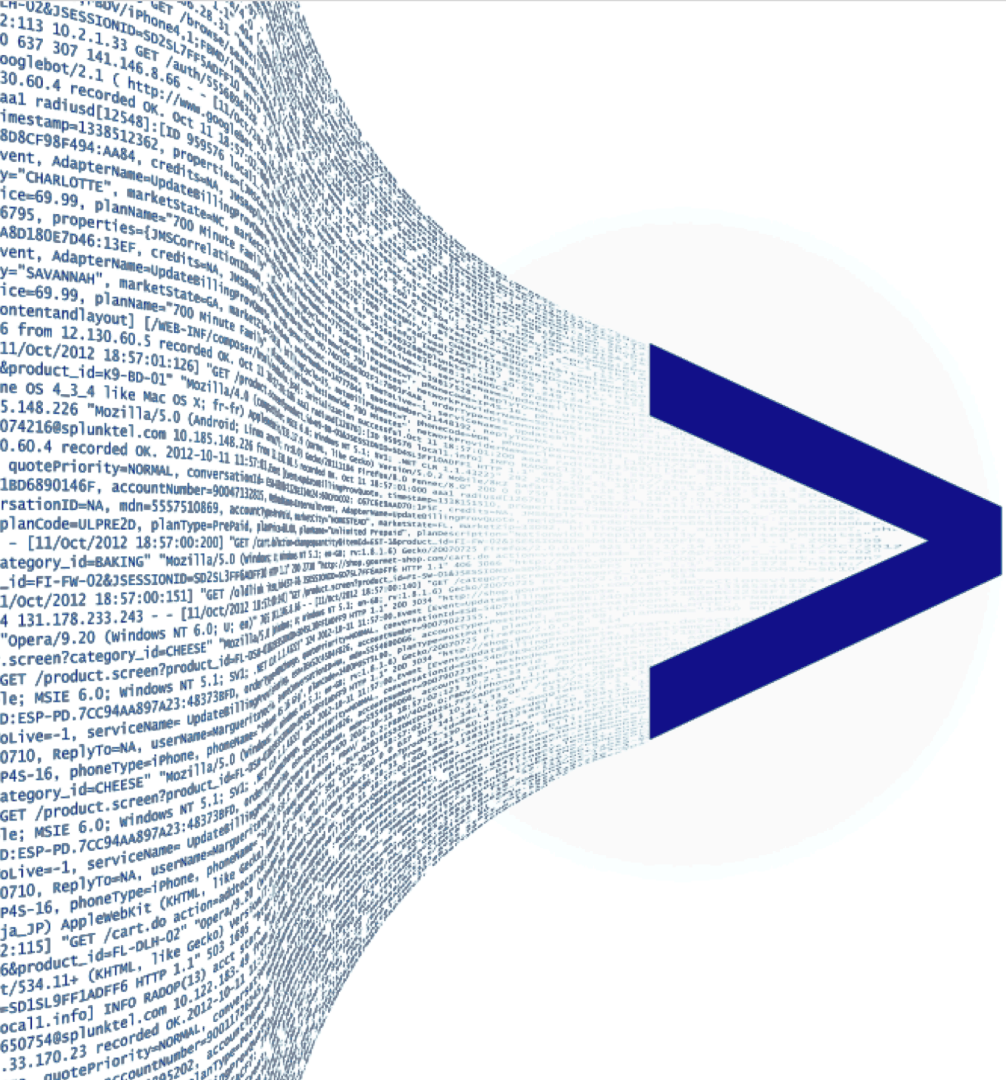
```
index=msexchange

|rename resolved_sender AS xsender

| stats  count by MID _time recipient sender xsender  subject

| stats list(_time) AS Date   list(recipient) AS Recipient  list(sender) AS PurportedSender
list(xsender) AS XSender list(subject) AS Subject  list(count) AS Count by MID

| search PurportedSender=*@company.com
| where  PurportedSender!=Xsender
| where  XSender!="<>"
```

You Can't Phish Me! – Next Level

- Some Splunk customers are correlating suspect messages with new process launches or other endpoint events

- Suspicious Messages:
  - New Senders
  - Unusual Domains
  - Lexical analysis of Email Body
    - Super advanced
  - "Warning" level in email security software

- Correlating Events:
  - New Process Launches
  - Increase in Data Transmission
  - AV Alerts
  - HIDS Alerts



Analytics Searches

You Uploaded What Where? – High Level Search

- index=isa tag=target_users tag=cloud_providers AND NOT tag=cloud_whitelist
  - For in scope users, look at their cloud uploads that aren't whitelisted
- | stats sum(eval(bytes_in+bytes_out)) as flow_sum, sum(bytes_out) as bytes_out_sum by user _time
  - Pull the total data folow, and the outbound flow per user per day
- | eventstats avg(flow_sum) as average, stdev(flow_sum) as stdev by user | eval threshold=stdev+average
  - Calculate the average and stdev per user, while, maintaining the full event history
- | where flow_sum>threshold AND bytes_out_sum/flow_sum >.5
  - Filter for users that are more than one standard deviation above normal, and where more than half the data was an upload vs a download
- | chart median(eval((flow_sum/threshold*100)-100)) by _time user
  - Create a usable chart of this data. This customer uses dashboards in addition to tickets

You Uploaded What Where? – Actual Search

```
index=isa tag=target_users tag=cloud_providers AND NOT
tag=cloud_whitelist  | bin span=1d _time | eval flow=bytes_in
+bytes_out   | stats sum(flow) as flow_sum, sum(bytes_out) as
bytes_out_sum  by user _time  | eventstats avg(flow_sum) as average,
stdev(flow_sum) as stdev by user | eval threshold=stdev+average  |
eval flow_ratio=bytes_out_sum/flow_sum  | where
flow_sum&gt;threshold AND flow_ratio&gt;.5  | eval
exceeds_threshold= flow_sum-threshold   | eval
perc_over_threshold=(flow_sum/threshold*100)-100  | chart
median(perc_over_threshold) by _time user
```

You Copied How Many USB Files? – High Level Search

- index=sep* api="File Write" tag=target_users `sep_write_exclude`
  - Pull non-whitelisted SEP data for target users
- | bin span=1d _time | stats count by user _time
  - Count the number of events per day
- | eventstats avg(count) as average, stdev(count) as stdev by user
  - Find average and stdev per user
- | where count>stdev+average
  - Find deviant users
- | chart median(eval((count/threshold*100)-100)) by _time user
  - Pull the deviation for deviant users over time

You Copied How Many USB Files? – Actual Search

```
index=sep* api="File Write"  tag=target_users
`sep_file_write_exclusions` `sep_time_adjustments` | eval
user=user_name | bin span=1d _time | stats count as count  by
user_name _time | eventstats avg(count) as average, stdev(count) as
stdev by user_name | eval threshold=stdev+average | where
count&amp;gt;threshold | eval exceeds_threshold=count-threshold  | eval
perc_over_threshold=(count/threshold*100)-100 | table _time
user_name sparkline threshold count exceeds_threshold
perc_over_threshold  | chart median(perc_over_threshold) by _time
user_name
```

Dashboard To Correlation Search (Easy Version)

- index=sep* api="File Write" tag=target_users `sep_write_exclude`
  - Pull non-whitelisted SEP data for target users
- | bin span=1d _time | stats count by user _time
  - Count the number of events per day
- | **stats latest(count) as latest** avg(count) as average, stdev(count) as stdev by user
  - Find average and stdev per user
- | where **latest**>stdev+average
  - Find deviant users

Dashboard To Correlation Search (Harder Version)

- index=sep* api="File Write" tag=target_users `sep_write_exclude`
- | bin span=1d _time | stats count by user _time
- | stats latest(count) as latest
  avg(eval(_time < relative_time(now(), "-1d”),count,null)) as average,
  stdev(eval(_time < relative_time(now(), "-1d”),count,null)) as stdev
  by user
  - Exclude the most recent day from the average and stdev
- | where latest>stdev+average

Generic Statistical Search Builde

- index=somedatasource
- | bin span=1d _time | stats count by user _time
- | stats latest(count) as latest
  avg(eval(_time < relative_time(now(), "-1d”),count,null)) as average,
  stdev(eval(_time < relative_time(now(), "-1d”),count,null)) as stdev
  by user
  - Exclude the most recent day from the average and stdev
- | where latest>stdev+average

Maybe host instead?

Maybe 2*stdev instead?

You Copied How Many USB Files? – Table Edition

```
index=sep* api="File Write"  tag=target_users  `sep_file_write_exclusions`
`sep_time_adjustments` | eval user=user_name | fields _time user_name |
chart sparkline count as count  by user_name | eventstats avg(count) as
average, stdev(count) as stdev | eval threshold=stdev+average | where
count&amp;gt;threshold | eval exceeds_threshold=count-threshold  | eval
perc_over_threshold=(count/threshold*100)-100 | table user_name sparkline
threshold count exceeds_threshold perc_over_threshold  | sort -
perc_over_threshold | fieldformat threshold = tostring(threshold, "commas")
| fieldformat count = tostring(count, "commas") | fieldformat
exceeds_threshold = tostring(exceeds_threshold, "commas") | fieldformat
perc_over_threshold = tostring(perc_over_threshold, "commas")
```

Detecting Wire Transfer Phishing – High Level Search

- index=bro_smtp (wire subject=*wire* (transfer subject=*transfer*) OR (funds subject=*funds*)) OR (remediate subject=*remediate*)) […]
  - Munge your data and do any field extractions
- | eval exclude=if(match(subject,"Wire Transfer Approval Required [A-Z]{2}\d{6}"),1,0) [….] | search NOT(mailfrom=*@bounce.secureserver.net […])
  - Exclude whitelisted / tuned sources
- | eval phish_score=1| eval phish_score=case(reply_to==from,phish_score,reply_to=="-",phish_score,1==1, (phish_score*10)+1) | eval phish_score=case(reply_to_domain==from_domain,phish_score,reply_to=="-",phish_score,1==1,(phish_score*10)+1) […]
  - Build threshold
- | where targeted_score>1000000 AND phish_score>10000000 AND spammy<24
  - Filter for highly anomalous emails

Detecting Wire Transfer Phishing – Actual Search

index=bro_smtp (("Workspace Webmail") OR ("Roundcube Webmail")) OR ((wire subject=*wire* ((transfer subject=*transfer*) OR (funds subject=*funds*))) OR (remediate subject=*remediate*) OR (remit* subject=*remit*)) | rex field=to mode=sed "s/(\"[^,]*),([^,]*\")/\1_\2/g"| makemv delim="," to| mvexpand to| rex field=mailfrom "\<(?<mailfrom>[^\>]+)\>"| rex field=sender "\<(?<sender>[^\>]+)\>"| rex field=recipient "\<(?<recipient>[^\>]+)\>"| rename sender as from, recipient as rcptto| rex field=to "\<(?<to>[^\>]+)\>"| regex to="@" | rex field=reply_to "\<(?<reply_to>[^\>]+)\>"| rex field=mailfrom "@(?<mailfrom_domain>.*)$"| rex field=mailfrom_domain "(?<mailfrom_basedomain>[^\.]+\.[^\.]+)$"| rex field=from "@(?<from_domain>.*)$" | rex field=from_domain "(?<from_basedomain>[^\.]+\.[^\.]+)$"| rex field=reply_to "@(?<reply_to_domain>.*)$"| rex field=reply_to_domain "(?<reply_to_basedomain>[^\.]+\.[^\.]+)$"| eval from=lower(from)| eval to=lower(to)| eval rcptto=lower(rcptto)| eval reply_to=lower(reply_to)| eval mailfrom=lower(mailfrom)

- Munge your data and do any field extractions

| eval exclude=if(match(subject,"Wire Transfer Approval Required [A-Z]{2}\d{6}"),1,0)| eval exclude=if(match(subject,"WIRE TRANSFER Notification:.*, File \#\d+, Expense Report \#\d+"),1,0)| eval exclude=if(match(subject,"Infoworks Notification \d{9}"),1,0)| eval exclude=if(match(subject,"[Gg]roupon"),1,0)| eval exclude=if(match(mailfrom,"[Gg]roupon"),1,0)| eval exclude=if(match(from,"[Gg]roupon"),1,0)| eval exclude=if(match(subject,"Delivery Status Notification (Failure)",1,0)| search (to=*[domain]* OR to=*[domain]*)AND NOT(mailfrom=*@bounce.secureserver.net OR mailfrom="dt.pixture@gmail.com" OR mailfrom=*@googlegroups.com OR mailfrom=postmaster@urs.com OR mailfrom=*@[domain].com OR mailfrom=*@mail.cignasecure.com OR mailfrom=*@[domain].com OR from=*@[domain].com OR from=*@[domain].com OR from="*.groupon.com" OR from="service@remitly.com" OR from=*schneider-electric.com OR mailfrom=*@[domain] OR mailfrom=*@brindyl.com OR mailfrom=*.[domain].com OR mailfrom=*@myfax.com OR mailfrom=*@observium.org OR mailfrom=*@xansys.org OR from=*@app.topica.com OR from=info@securemailingtoday.com OR to=*@[domain].com OR to=agencies@[domain].com OR to=observium@observium.org OR (reply_to="-" AND mailfrom="<>" AND (from=*[domain].com OR from=*[domain].com)) OR (exclude=1 AND (to=*[domain].com OR to=*[domain].com) AND (from=*[domain].com OR from=*[domain].com))) AND NOT ("you're pre-approved" OR "chance to win" OR "your preapproved" OR "your pre-approved" OR "splunk alert" OR "Delivery Status Notification (Failure)")

- Exclude whitelisted / tuned sources

| eval phish_score=1| eval phish_score=case(reply_to==from,phish_score,reply_to=="-",phish_score,1==1,(phish_score*10)+1) | eval phish_score=case(reply_to_domain==from_domain,phish_score,reply_to=="-",phish_score,1==1,(phish_score*10)+1) | eval phish_score=case(reply_to_basedomain==from_basedomain,phish_score,reply_to=="-",phish_score,1==1,(phish_score*10)+1) | eval phish_score=if(mailfrom_domain==from_domain,phish_score,(phish_score*10)+1) | eval phish_score=if(mailfrom_basedomain==from_basedomain,phish_score,(phish_score*10)+1) | eval phish_score=if(mailfrom==from,phish_score,(phish_score*10)+1)| eval phish_score=if(match(mailfrom_domain,"{closer namealike pattern1}"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(mailfrom_domain,"{closer namealike pattern1}"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(mailfrom_domain,"{looser namealike pattern3}"),(phish_score*10)+1,phish_score)| eval phish_score=if(match(mailfrom_domain,"{closer namealike pattern4}"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(mailfrom_domain,"{closer namealike pattern5}"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(mailfrom_domain,"{looser namealike pattern6}"),(phish_score*10)+1,phish_score)| eval phish_score=if(match(mailfrom_domain,"{closer namealike pattern7}"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(mailfrom_domain,"{closer namealike pattern8}"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(mailfrom_domain,"{looser namealike pattern9}"),(phish_score*10)+1,phish_score)| eval phish_score=if(match(from_domain,"{closer namealike pattern1}"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(from_domain,"{closer namealike pattern2}"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(from_domain,"{looser namealike pattern3}"),(phish_score*10)+1,phish_score)| eval phish_score=if(match(from_domain,"{closer namealike pattern4}"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(from_domain,"{closer namealike pattern5}"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(from_domain,"{looser namealike pattern6}"),(phish_score*10)+1,phish_score)| eval phish_score=if(match(from_domain,"{closer namealike pattern7}"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(from_domain,"{closer namealike pattern8}"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(from_domain,"{looser namealike pattern9}"),(phish_score*10)+1,phish_score)| eval phish_score=if(match(from_domain,"(yahoo|gmail|outlook|hotmail|freemail|hushmail|googlemail|live\.com|msn\.com)"),(phish_score*10)+1,phish_score)| eval phish_score=if(match(reply_to_domain,"(yahoo|gmail|outlook|hotmail|freemail|hushmail|googlemail|live\.com|msn\.com)"),(phish_score*10)+1,phish_score)| eventstats count(subject) as spammy by subject| eval targeted_score=phish_score*(1/spammy)

- Build threshold

| where targeted_score>1000000 AND phish_score>10000000 AND spammy<24| table _time mailfrom from reply_to rcptto to subject user_agent spammy phish_score targeted_score| sort -targeted_score, subject, -phish_score, mailfrom, _time

- Filter out malicious content

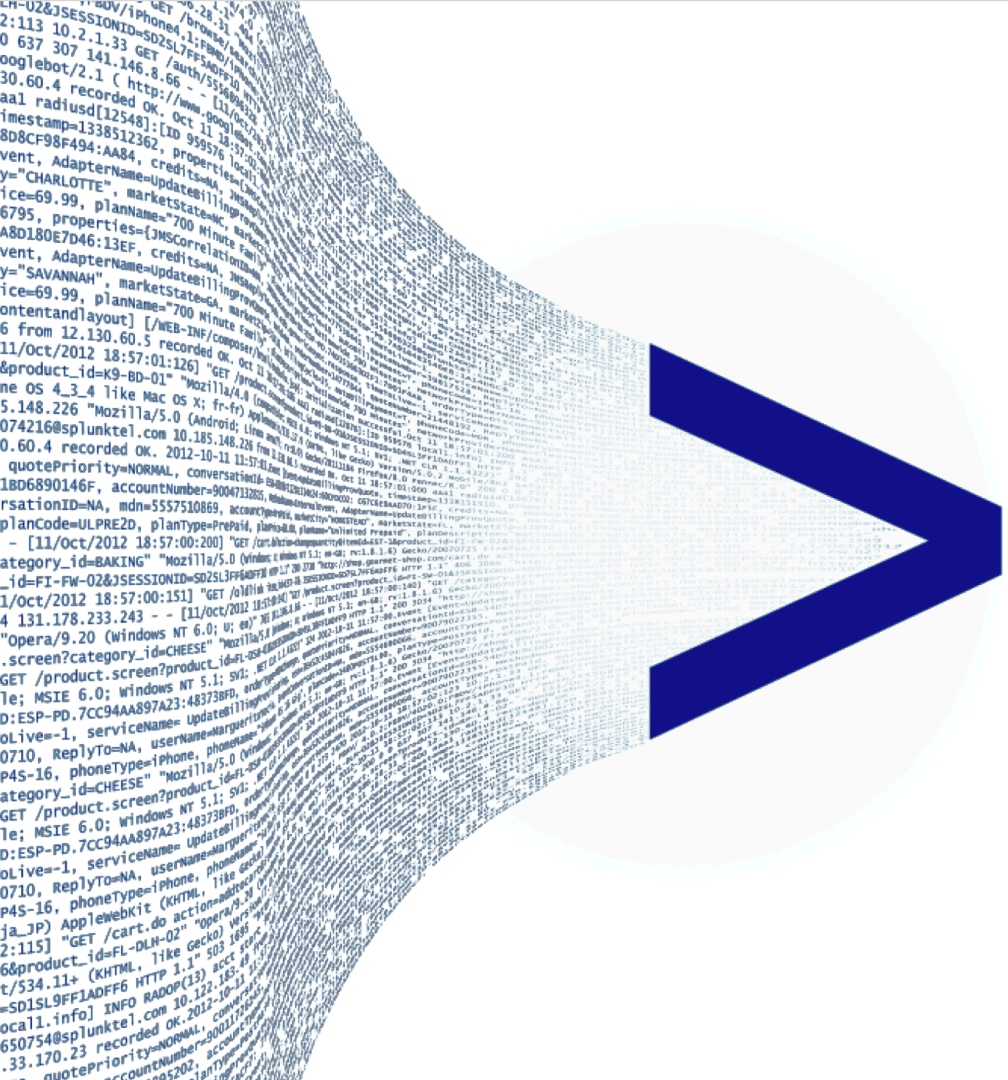
- This search, printed out in its formatted form, is 101 lines long. While the logic is highly developed and tuned, it's functionally no different than any other search.

Expired User Activity – Search

```
index=win* OR index=nix OR index=dpalm (src_user_endDate=* OR user_endDate=*) | fillnull
value="-"
| eval "src_user_endDate” = case(isnum('src_user_endDate'),'src_user_endDate',
isnum(strptime('src_user_endDate',"%m/%d/%Y")),strptime('src_user_endDate',"%m/%d/
%Y"),isnum(strptime('src_user_endDate',"%m/%d/%y")),strptime('src_user_endDate',"%m/%d/%y"),
1=1,'src_user_endDate')
| eval "user_endDate” = case(isnum('user_endDate'),'user_endDate',
isnum(strptime('user_endDate',"%m/%d/%Y")),strptime('user_endDate',"%m/%d/
%Y"),isnum(strptime('user_endDate',"%m/%d/%y")),strptime('user_endDate',"%m/%d/%y"),
1=1,'user_endDate')
| eval expired_user = case(isnotnull(user_endDate),user, isnotnull(src_user_endDate),src_user,
1=1,null())
| eval endDate=strftime(user_endDate,"%Y-%m-%d")
| where endDate>=user_startDate
| stats count by  sourcetype  user src src_ip name  endDate user_startDate
| stats values(sourcetype) AS sourcetype list(src) AS Source  list(src_ip) AS SrcIP list(name) AS Message
values(endDate) AS EndDate values(user_startDate) AS StartDate list(count) AS count by user

| `mvdedup(sourcetype)` | `mvdedup(EndDate)` | `mvdedup(StartDate)`
```

Get An Angle On Angler – Actual Search

- daysago=7 (index=bro_http OR index=[proxy logs] OR index=[firewall logs])
  - [search daysago=7 (index=bro_http OR index=[proxy logs] OR index=[firewall logs]) ("viewforum.php" OR "index.php" OR "search.php" OR "viewtopic.php" OR "/topic/")
  - | regex url="(\.php\?(PHPSESSID|keywords|t|f)=[a-z\.0-9]+&(f|sid|action|fid0)=[a-z\.&0-9]{10,31}$|\/topic\/[0-9]{5}(\-[a-z]+){7}(\/$|$))" | rex field=domain "^.+?\.(?P<sub_domain>.+?)\." | dedup sub_domain | fields sub_domain | rename sub_domain as search | format]
- | rex field=domain "^.+?\.(?P<sub_domain>.+?)\."
- | rex field=url "^.+\/(?P<URI>[^\/]+?)$"
- | bin _time span=1h | eventstats dc(URL) AS COUNT by domain, _time | eventstats dc(domain) AS D_COUNT by sub_domain, _time
- | where ((COUNT > 2) OR (D_COUNT > 1))
- | table _time, sub_domain, domain, URI



General SOC/HR Searches

I Saw That Query, Bing Edition – Search

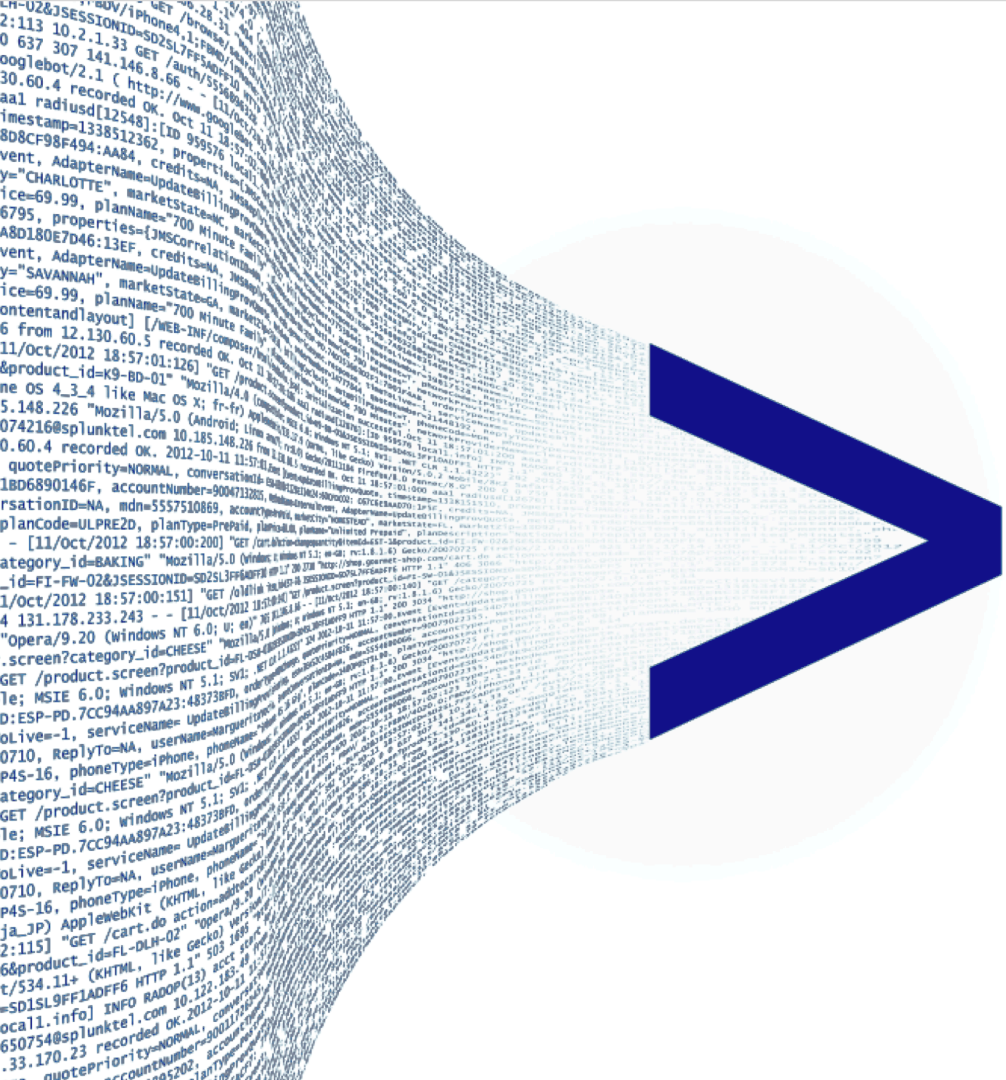
- index=proxy sourcetype=bluecoat "http://www.bing.com/search*"
- | rex field=_raw "^[^=\\n]*=(?P<search>[^&]+)"
- | stats count by user search action
- | where len(user)>1
- | sort -count
- | stats list(user) as "User", list(count) as Count by search action

I Saw That Query, Google Edition – Search

- index=proxy sourcetype=bluecoat "Google Search"
- | rex field=_raw "^(?:[^=\\n]*=){4}(?P<search>[a-z]+)"
- | stats count by user search action
- | where len(search)>1
- | sort -count
- | stats list(search) as "Search", list(count) as Count by user action

I Saw That Video, Youtube Edition – Search

```
sourcetype=bluecoat  dest="*youtube.com*"  cs_uri_query="*?search_query=*"
| eval subject=lower(subject)
| eval cs_uri_query=replace(cs_uri_query,"^.*(search_query=)","")
| eval cs_uri_query=split(cs_uri_query,"&spf=navigate")
| rex field=cs_uri_query "(?<query>^[^\+]+(\+[^\+]+)*$)"
| eval query=replace(query,"\+"," ")
| table _time user query cs_uri_query
```



Conclusion

Thank You To Contributors

THANK YOU

splunk>