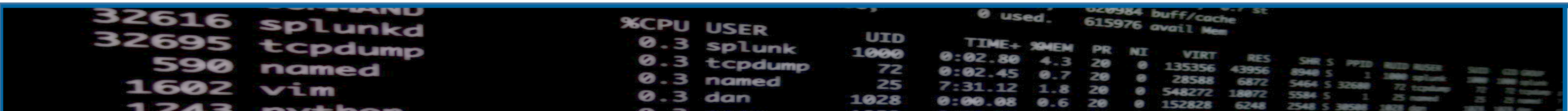
Splunk and Windows Event Log: Best Practices, Reduction and Enhancement

David Shpritz

Aplura, LLC

Baltimore Area Splunk User Group June 2017



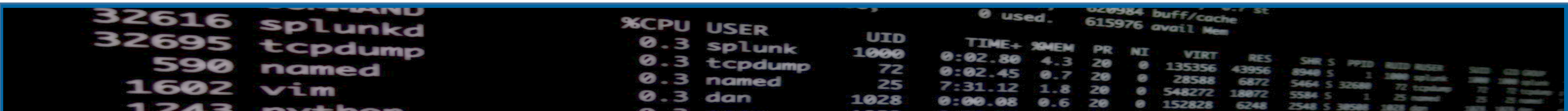
```
32616  splunkd      %CPU USER      UID        TIME+  %MEM  PR  NI
32695  tcpdump       0.3 splunk   1000    0:02.80   4.3  20   0
  590  named         0.3 tcpdump    72    0:02.45   0.7  20   0
 1602  vim           0.3 named      25    7:31.12   1.8  20   0
                     0.3 dan      1028    0:00.08   0.6  20   0
```

APLURA

Many Solutions, One Goal.

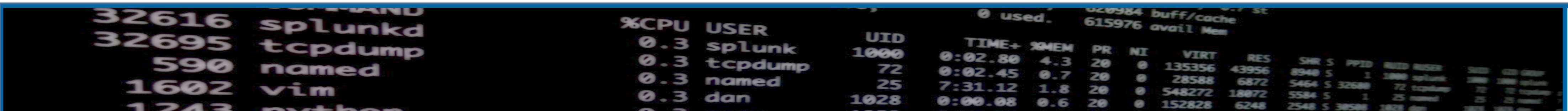
Agenda

- Getting Windows Events into Splunk: Patterns and Practices
- TURN DOWN THE VOLUME: License reduction tips
- Making them more useful: Improving knowledge objects

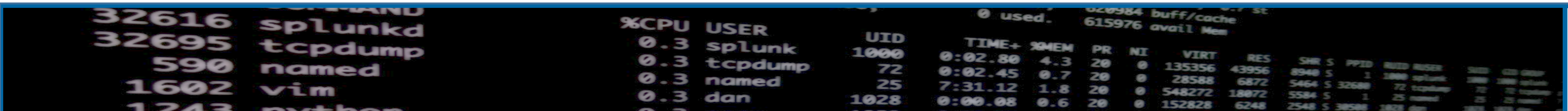


Ground Rules

- Fidelity levels
    - How complete are the events?
- Windows Event interpretation
    - These are binary records
    - Agents can read them directly or ask the Windows API
    - This means that you aren’t really getting the event log, just a representation of it



Getting Windows Events into Splunk

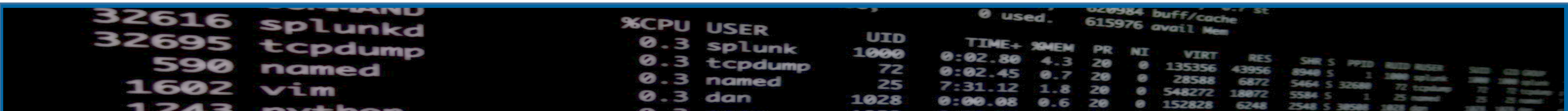


Different Ways to Skin a Cat

- Best to Worst
  - Universal Forwarder
  - Windows Event Forwarding
  - WMI
  - EVTX Import
  - Third Party Syslog Agent (Snare, for example)

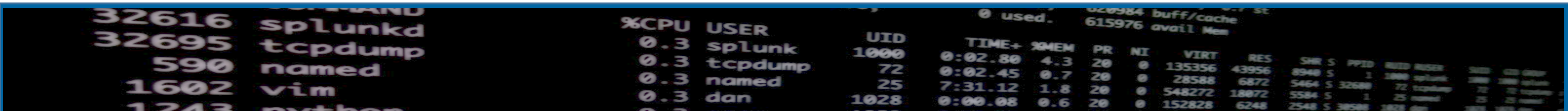
Universal Forwarder

- The best way to get Windows events (of course we’re biased)
- Pros
    - High fidelity
    - Can be controlled by Deployment Server
    - Can filter Windows events
    - Can run scripts (batch, exe, PS)
    - Can also get admon (great for assets and identities)
- Cons
    - “Another agent!?!?”
    - Security concerns



Windows Event Forwarding

- Native to Windows (2008R2 and up)
- Pros
  - Native to Windows, no agent
  - Can be configured with GPO
- Cons
  - Almost high fidelity
  - Slower
  - Scalability issues
  - Customer testing shows it consumes more resources than a UF

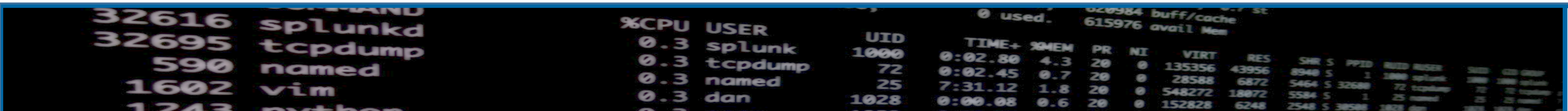


```
32616  splunkd        %CPU USER       UID     TIME+   %MEM  PR  NI    VIRT
32695  tcpdump         0.3 splunk    1000   0:02.80   4.3  20   0  135356
  590  named           0.3 tcpdump     72   0:02.45   0.7  20   0   28588
 1602  vim             0.3 named       25   7:31.12   1.8  20   0  548272
                       0.3 dan       1028   0:00.08   0.6  20   0  152828
```

Many Solutions, One Goal.

WMI

- Used by a Splunk system to collect Windows Events from a remote system
- Pros
  - Remote, no agent
- Cons
  - Slow
  - A lot of overhead
  - Limited collection availability (may need multiple systems to pull all your Windows hosts)
  - Low fidelity
  - Dealing with permissions



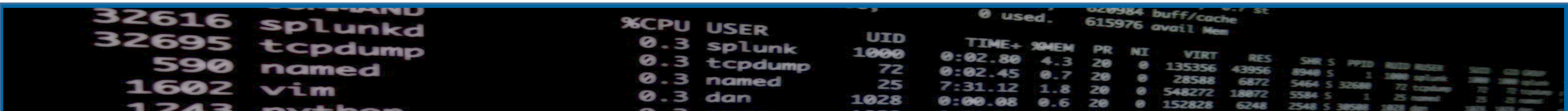
Many Solutions, One Goal.

EVTX Import

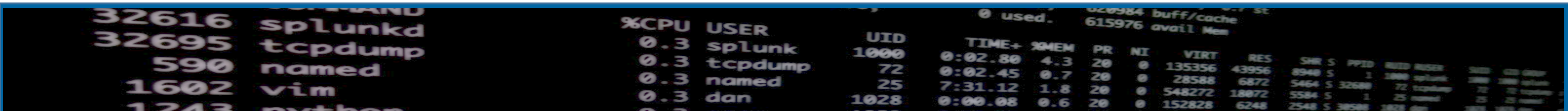
- Can be used to export event logs from a system and then import the raw files on another system
- Often seen in ”air-gapped” environments
- Pros
  - No network connection needed from the client systems to the target indexers
- Cons
  - Low fidelity (remember that “interpretation” thing earlier?)
  - Moving and removing the files is a manual process
  - Open to event duplication

Third Party Syslog Agent (Snare)

- It's a thing, these agents exist
- Pros
  - Can work with your existing syslog infrastructure
- Cons
  - Super low fidelity
  - Unreliable (syslog never dies)
  - Remote configuration?



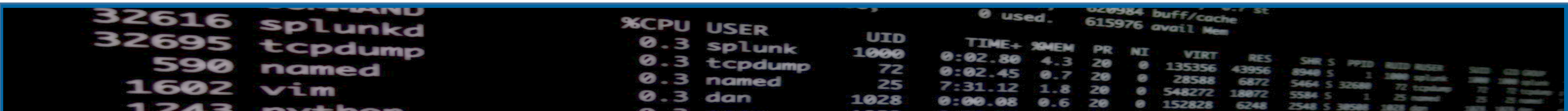
TURN DOWN THE VOLUME: License reduction tips



```
                          0 used.     620984 buff/cache
                                      615976 avail Mem
32616 splunkd    %CPU USER    UID    TIME+  %MEM  PR  NI
32695 tcpdump     0.3 splunk  1000  0:02.80  4.3  20   0
  590 named       0.3 tcpdump   72  0:02.45  0.7  20   0
 1602 vim         0.3 named     25  7:31.12  1.8  20   0
                  0.3 dan     1028  0:00.08  0.6  20   0
```

These things are chatty

- Splunk estimates between 200-300mb per day, per system
- Of course, that can vary wildly
- Lots of repeated events with little to no value (looking at you 4662)
- Do we really need all of these?
- Do we need every part of all of these?

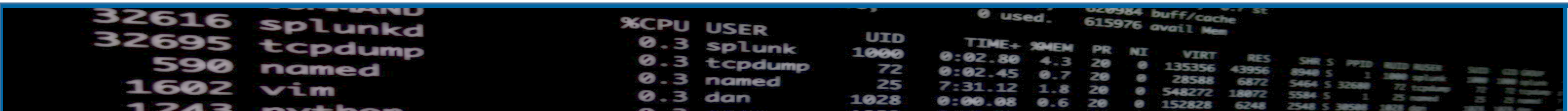


```
32616 splunkd      %CPU USER        UID    TIME+  %MEM  PR  NI    VIRT    RES
32695 tcpdump       0.3 splunk     1000  0:02.80  4.3  20   0  135356  43956
  590 named         0.3 tcpdump      72  0:02.45  0.7  20   0   28588   6872
 1602 vim           0.3 named        25  7:31.12  1.8  20   0  548272  18072
                    0.3 dan        1028  0:00.08  0.6  20   0  152828   6248
```

Many Solutions, One Goal.

Stratergery

- Pick your systems carefully
- Pick your inputs carefully on those systems
- Whitelist and Blacklist carefully
- Resolving objects
- Baseline?
- Current_only? Start_from?
- XmlWinEventLog
- Filtering and cleaning up

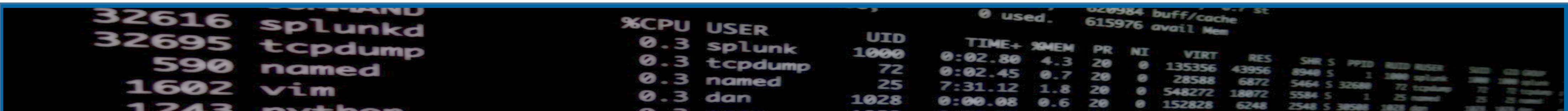


```
32616 splunkd     %CPU USER       UID   TIME+    %MEM  PR  NI
32695 tcpdump      0.3 splunk    1000  0:02.80   4.3  20   0
  590 named        0.3 tcpdump     72  0:02.45   0.7  20   0
 1602 vim          0.3 named       25  7:31.12   1.8  20   0
                   0.3 dan       1028  0:00.08   0.6  20   0
```

Many Solutions, One Goal.

Which systems?

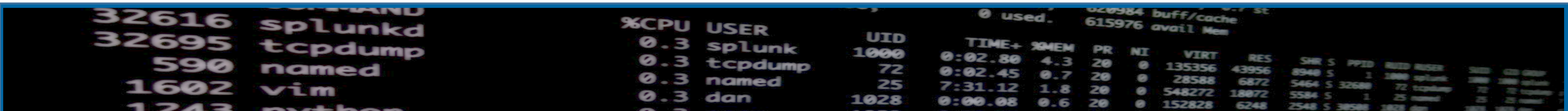
- Just Active Directory servers?
- Endpoints?
- Servers?
- Sorry, this is on a case by case basis



```
32616 splunkd      %CPU USER     UID
32695 tcpdump       0.3 splunk   1000
  590 named         0.3 tcpdump    72
 1602 vim           0.3 named      25
                    0.3 dan      1028
```

Picking your inputs (not your nose)

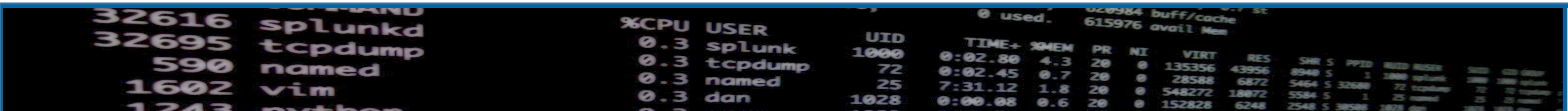
- Set a baseline for which logs ALL your systems should be sending
- For other eventlogs, use an individual app for turning on that input (DS-Input-wineventlog_application)
- Do you need admon from all your systems? Probably not, just on a few AD systems
- Make sure you aren't using legacy inputs (WMI vs Perfmon)
- Look out for Windows Firewall Events (maybe Stream instead?)



```
32616  splunkd        %CPU USER
32695  tcpdump         0.3 splunk
  590  named           0.3 tcpdump
 1602  vim             0.3 named
                       0.3 dan
```

Whitelisting and Blacklisting

- Can have a big impact on your license usage
- Investing the time in “which events” can pay off big
- Careful with a whitelist-only approach
- Note that there is a limit to the number of lists
- Performed at the forwarder, so does not use network traffic

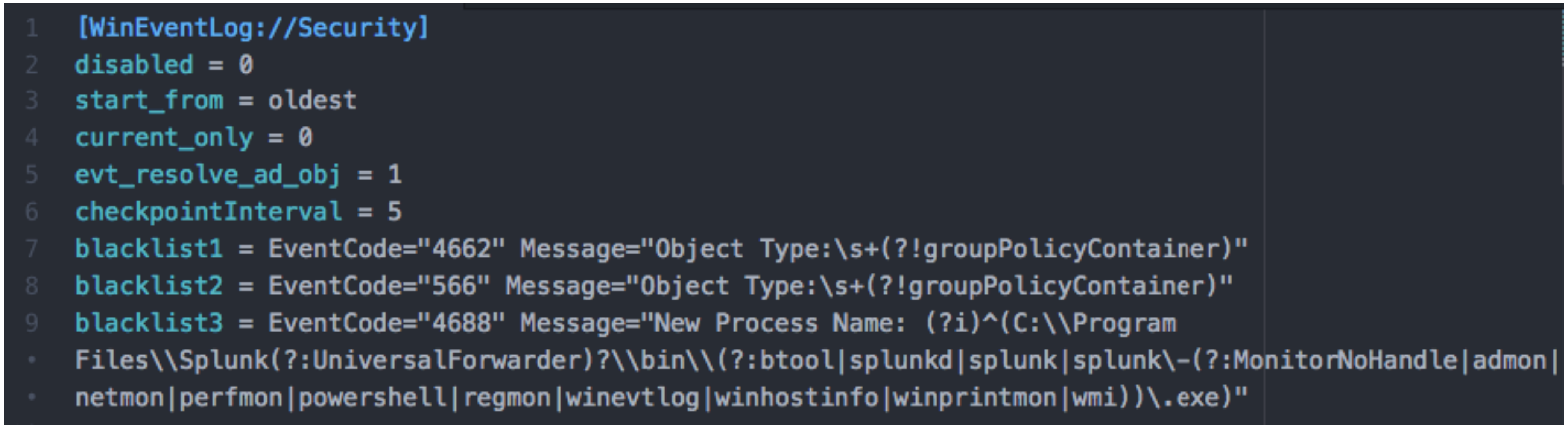


```
                                                 0 used.      615976 avail Mem
32616  splunkd        %CPU USER          UID    TIME+    %MEM  PR  NI    VIRT    RES
32695  tcpdump         0.3 splunk       1000    0:02.80   4.3  20   0  135356  43956
  590  named           0.3 tcpdump        72    0:02.45   0.7  20   0   28588   6872
 1602  vim             0.3 named          25    7:31.12   1.8  20   0  548272  18072
                       0.3 dan          1028    0:00.08   0.6  20   0  152828   6248
```

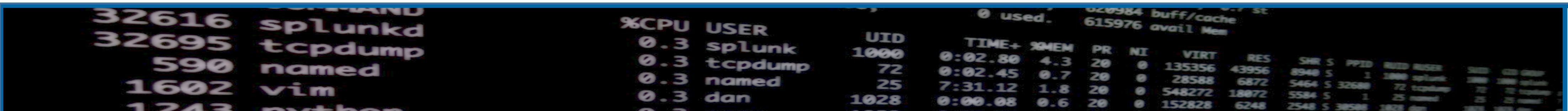
Many Solutions, One Goal.

Some nice blacklist options to start with

- https://gist.github.com/automine/a3915d5238e2967c8d44b0ebcfb66147



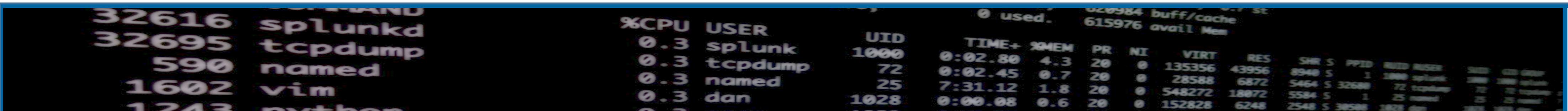
```
[WinEventLog://Security]
disabled = 0
start_from = oldest
current_only = 0
evt_resolve_ad_obj = 1
checkpointInterval = 5
blacklist1 = EventCode="4662" Message="Object Type:\s+(?!groupPolicyContainer)"
blacklist2 = EventCode="566" Message="Object Type:\s+(?!groupPolicyContainer)"
blacklist3 = EventCode="4688" Message="New Process Name: (?i)^(C:\\Program
Files\\Splunk(?:UniversalForwarder)?\\bin\\(?:btool|splunkd|splunk|splunk\-(?:MonitorNoHandle|admon|
netmon|perfmon|powershell|regmon|winevtlog|winhostinfo|winprintmon|wmi))\.exe)"
```



Many Solutions, One Goal.

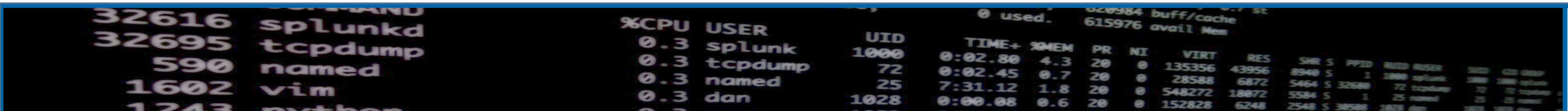
AD Object Resolution

- Resolves things like SIDs and GIUDs
- You can tell Splunk which DCs to use to resolve these
- Can add some overhead (CPU and Memory), but usually low impact
- Recommendation is to resolve them (look at the evt_*) options in inputs.conf for Windows Event Logs



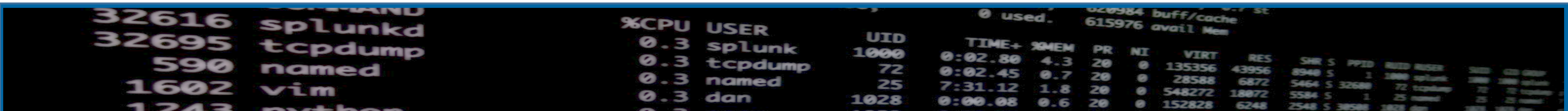
Baselining AD

- Will collect your whole AD schema
- Can take up a lot of memory on AD controllers
- But baselining is useful for Assets and Identities in ES
- So be careful which systems you baseline on



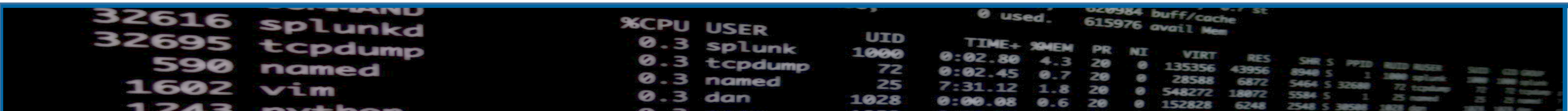
Current_only vs. start_from

- Current_only tells Splunk to only grab the latest events (like tail –f, if Windows had such a thing)
- Useful to make sure you don’t get all the historical data
- May want to set that to “true” on initial deployment
- Then set to “false”, restart, and it should pick up from the checkpoint
- Start_from should be “oldest”
- Setting it to “newest” can be used to grab a backlog of events
  - I’ve never seen this in the wild



XmlWinEventLog

- Should reduce license usage (claims are up to 70%)
- It will always be in English (pro? Con?)
- Harder to read, I mean, it’s XML
- Quality of CIM compliance has been varied in the past
- It doesn’t ”look like Windows events” and some auditors are not bright
- What if you could get the same log savings and the readability

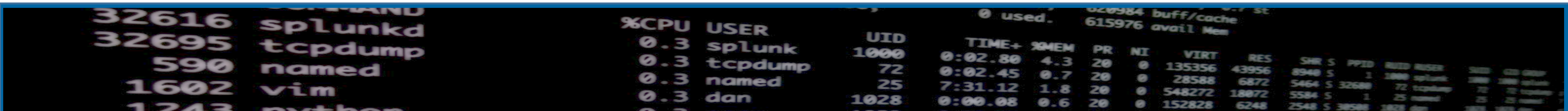


```
32616 splunkd      %CPU USER        UID
32695 tcpdump       0.3 splunk     1000
  590 named         0.3 tcpdump      72
 1602 vim           0.3 named        25
                    0.3 dan        1028
```

APLURA
Many Solutions, One Goal.

Filtering and cleaning up

- Don’t use “suppress_text”
- It’s tempting, but there goes the baby with the bathwater
- Maybe just clean up the text you don’t need

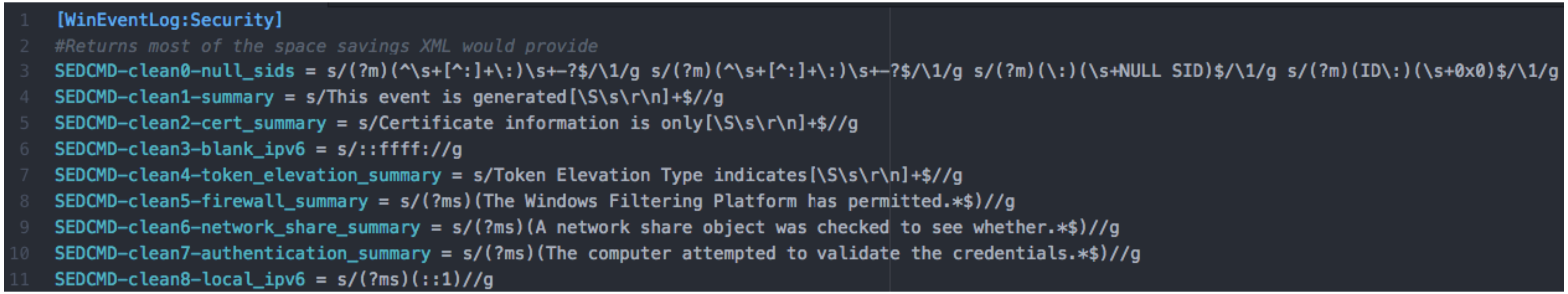


APLURA

Many Solutions, One Goal.

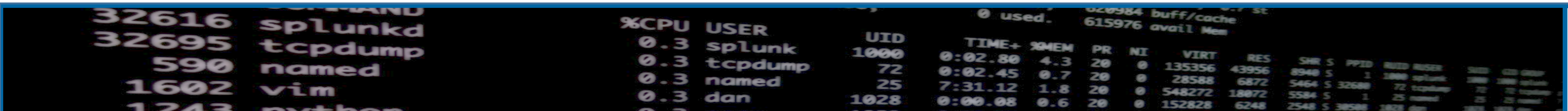
Filtering and cleaning up

- IPv6 support in event logs results in a lot of "::" and "ffff" and other garbage
- Let's clean up a lot (thanks to a lot of people for this)

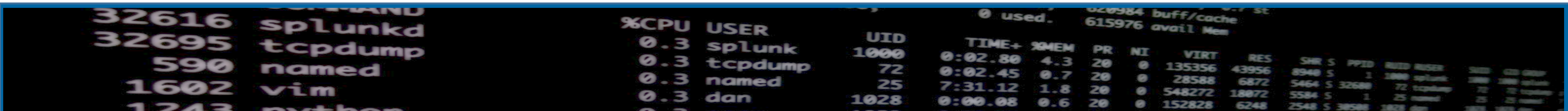


```
[WinEventLog:Security]
#Returns most of the space savings XML would provide
SEDCMD-clean0-null_sids = s/(?m)(^\s+[^:]+\:)\s+-?$/\1/g s/(?m)(^\s+[^:]+\:)\s+-?$/\1/g s/(?m)(\:)(\s+NULL SID)$/\1/g s/(?m)(ID\:)(\s+0x0)$/\1/g
SEDCMD-clean1-summary = s/This event is generated[\S\s\r\n]+$//g
SEDCMD-clean2-cert_summary = s/Certificate information is only[\S\s\r\n]+$//g
SEDCMD-clean3-blank_ipv6 = s/::ffff://g
SEDCMD-clean4-token_elevation_summary = s/Token Elevation Type indicates[\S\s\r\n]+$//g
SEDCMD-clean5-firewall_summary = s/(?ms)(The Windows Filtering Platform has permitted.*$)//g
SEDCMD-clean6-network_share_summary = s/(?ms)(A network share object was checked to see whether.*$)//g
SEDCMD-clean7-authentication_summary = s/(?ms)(The computer attempted to validate the credentials.*$)//g
SEDCMD-clean8-local_ipv6 = s/(?ms)(::1)//g
```

- https://gist.github.com/automine/5c8ef5b50e1df38249dfba01a70f2875



Making Them More Useful


```
32616 splunkd      %CPU USER        UID    TIME+  %MEM
32695 tcpdump       0.3 splunk     1000  0:02.80   4.3
  590 named         0.3 tcpdump      72  0:02.45   0.7
 1602 vim           0.3 named        25  7:31.12   1.8
                    0.3 dan        1028  0:00.08   0.6
```

Sorry, I ran out of time

- Got ES? Take a look at Ryan Faircloth's SecKit work
  - https://splunkbase.splunk.com/app/3059/
  - https://bitbucket.org/SPLServices/seckit_sa_idm_windows

- Alternative TAs
  - Should help with KO overhead
  - https://github.com/my2ndhead/TA-microsoft-windows (can do XML events)
  - https://bitbucket.org/SPLServices/seckit_ta_microsoft_windows (for use with SecKit)