

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

David Shpritz

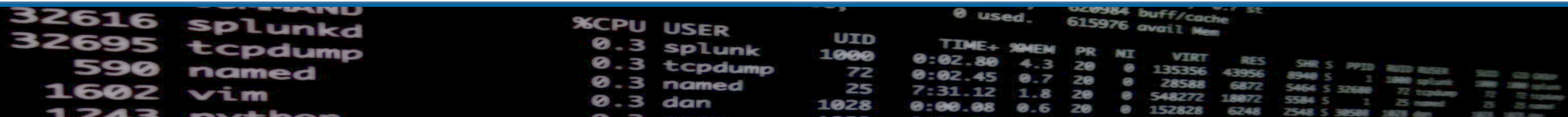
Aplura, LLC

Baltimore Area Splunk User Group June 2017

32616	splunkd	%CPU	USER	UID	TIME+	PMEM	PR	NI	VIRT	RES	SHR	S	PPID	PPID	USER	MEM	GROUP
32695	tcpdump	0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	3880	splunk	3880	splunk
590	named	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	72	tcpdump	72	tcpdump
1602	vim	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25	named	25	named
1243	python	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38580	1028	dan	1028	dan

Agenda

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

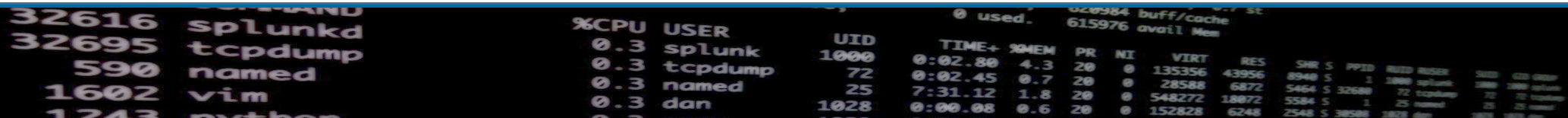


A terminal window screenshot showing system resource usage and a list of running processes. The top part of the terminal displays memory usage statistics, including '020984 buff/cache' and '615976 avail Mem'. Below this, a table lists system processes with columns for PID, CPU usage, user, UID, TIME+, MEM, PR, NI, VIRT, RES, SHR, S, PPID, and other details. The processes listed include 'splunkd', 'tcpdump', 'named', 'vim', and 'python'.

PID	%CPU	USER	UID	TIME+	MEM	PR	NI	VIRT	RES	SHR	S	PPID	OTHER
32616	0.3	splunkd	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	named
32695	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	72
590	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25
1602	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	30580	1602

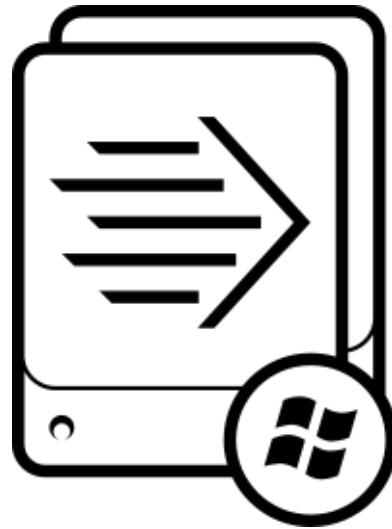
Ground Rules

- 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



A terminal window screenshot showing system information. On the left, a list of processes with their PIDs and names: 32616 splunkd, 32695 tcpdump, 590 named, 1602 vim, and 1243 python. On the right, a table of system statistics including CPU usage, user, UID, time, memory, and other metrics.

%CPU	USER	UID	TIME+	MEM	PR	NI	VIRT	RES	SHR	S	PPID	PPID	USER	MEM	GROUP
0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	3888	splunk	1880	splunk
0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	72	tcpdump	72	tcpdump
0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25	named	25	named
0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38584	1028	dan	1028	dan



Getting Windows Events into Splunk

[illegible]

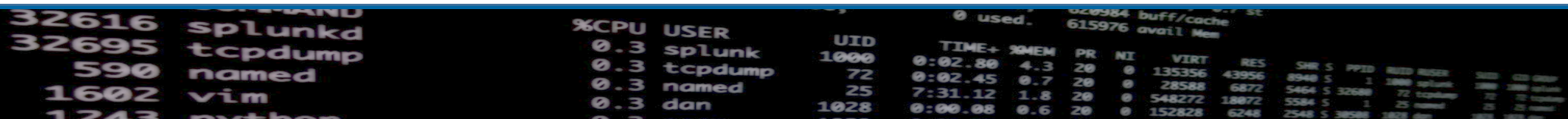
Different Ways to Skin a Cat

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

[illegible]

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



	%CPU	USER	UID	TIME+	PMEM	PR	NI	VIRT	RES	SHR	S	PPID	PID	NAME	GROUP
32616		splunkd													
32695		tcpdump													
590		named													
1602		vim													
1243		python													

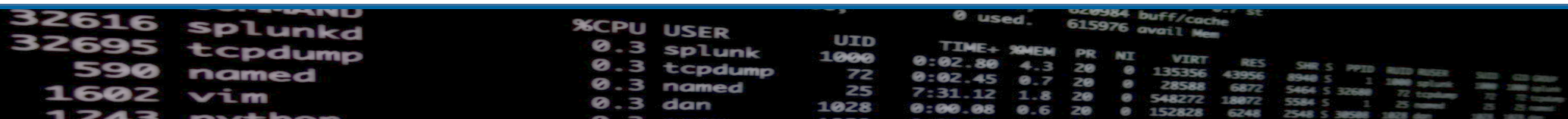
Windows Event Forwarding

- 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

	%CPU	USER	UID	TIME+	XMEM	PR	NI	VIRT	RES	SHR	S	PPID	RUID	RUSER	STAT	CPU	GROUP
32616 splunkd	0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	1000	splunk	1000	1000	splunk
32695 tcpdump	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	72	tcpdump	72	72	tcpdump
590 named	0.3	named	25	7:31.12	1.8	20	0	548272	18072	5584	S	1	25	named	25	25	named
1602 vim	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	30500	1028	dan	1028	1028	dan
1243 python	0.3	python	1028	0:00.08	0.6	20	0	152828	6248	2548	S	30500	1028	dan	1028	1028	dan

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

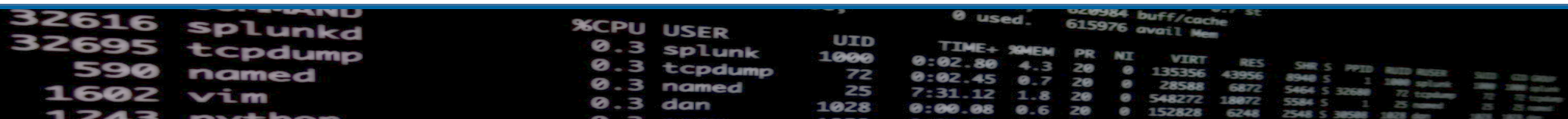


A terminal window screenshot showing system metrics and a process list. The top part displays memory usage: 0 used, 620984 buff/cache, 615976 avail Mem. Below this is a table with columns: %CPU, USER, UID, TIME+, %MEM, PR, NI, VIRT, RES, SHR, S, PPID, RSS, RSIZE, RTEXT, RDATA, RCODE, RSTACK, RSHR, RCODE, RDATA, RCODE, RSTACK. The table lists processes: splunkd (0.3% CPU, UID 1000, 0:02.80 TIME+, 4.3% MEM), tcpdump (0.3% CPU, UID 72, 0:02.45 TIME+, 0.7% MEM), named (0.3% CPU, UID 25, 7:31.12 TIME+, 1.8% MEM), and dan (0.3% CPU, UID 1028, 0:00.08 TIME+, 0.6% MEM).

%CPU	USER	UID	TIME+	%MEM	PR	NI	VIRT	RES	SHR	S	PPID	RSS	RSIZE	RTEXT	RDATA	RCODE	RSTACK	RSHR	RCODE	RDATA	RCODE	RSTACK
0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	2880	43956	2880	2880	43956	2880	43956	2880	43956	2880	43956
0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	72	tcpdump	72	72	tcpdump	72	72	tcpdump	72	72	tcpdump
0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25	named	25	25	named	25	25	named	25	25	named
0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38580	1828	dan	1828	1828	dan	1828	1828	dan	1828	1828	dan

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

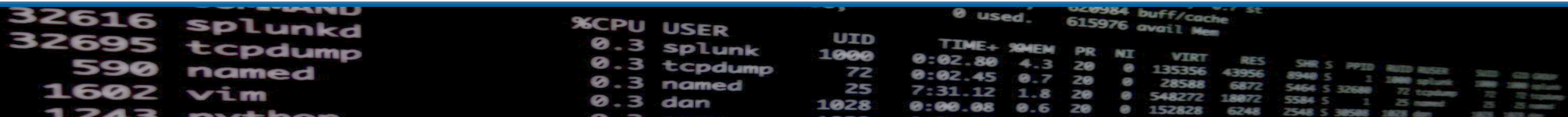


A terminal window screenshot showing system status and a list of processes. The top part shows memory usage: 0 used, 620984 buff/cache, 615976 avail Mem. Below is a table of processes with columns for PID, COMMAND, %CPU, USER, UID, TIME+, MEM, PR, NI, VIRT, RES, SHR, S, PPID, and others. The processes listed are splunkd, tcpdump, named, vim, and python.

PID	COMMAND	%CPU	USER	UID	TIME+	MEM	PR	NI	VIRT	RES	SHR	S	PPID	OTHERS
32616	splunkd	0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	2880
32695	tcpdump	0.3	splunk	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	72
590	named	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25
1602	vim	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38580	1628
1243	python	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38580	1628

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?



A terminal window screenshot showing system statistics and a list of processes. The top part of the terminal displays memory usage: 020984 buff/cache and 615976 avail Mem. Below this, a table lists processes with columns for PID, CPU usage, user, UID, TIME+, MEM, PR, NI, VIRT, RES, SHR, S, PPID, and other details. The processes listed include splunkd, tcpdump, named, vim, and python.

PID	%CPU	USER	UID	TIME+	MEM	PR	NI	VIRT	RES	SHR	S	PPID	OTHER
32616	0.3	splunkd	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	named
32695	0.3	splunk	1000	0:02.45	0.7	20	0	28588	6872	5464	S	72	tcpdump
590	0.3	named	72	7:31.12	1.8	20	0	548272	18872	5584	S	1	25 named
1602	0.3	vim	25	0:00.08	0.6	20	0	152828	6248	2548	S	30584	1828 vim
1243	0.3	python	1028										



TURN DOWN THE VOLUME:

License reduction tips

		%CPU		USER	UID	TIME+	MEM	PR	NI	VIRT	RES	SHR	S	PPID	PPID	USER	GROUP
32616	splunkd	0.3		splunk	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	2880	splunk	splunk
32695	tcpdump	0.3		tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	72	tcpdump	tcpdump
590	named	0.3		named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25	named	named
1602	vim	0.3		dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38580	1602	vim	vim
1243	python	0.3															

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?

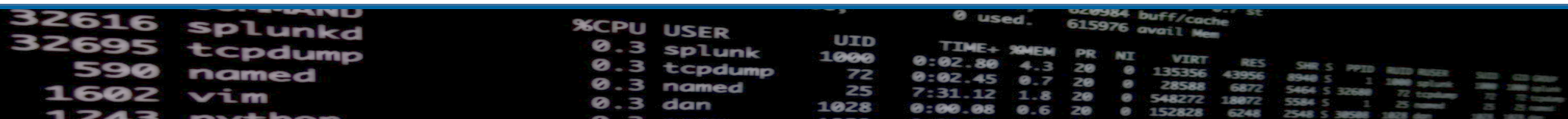


```
32616 splunkd
32695 tcpdump
590 named
1602 vim
1243 python
```

```
%CPU USER      UID      TIME+  %MEM  PR  NI  VIRT  RES  SHR  S  PPID  RSS  RSSD  RSSM  RSSV  RSSD  RSSM  RSSV
0.3 splunk    1000    0:02.80  4.3   20   0  135356 43956 8940 S   1  1000  1000  1000  1000  1000  1000  1000  1000  1000
0.3 tcpdump   72      0:02.45  0.7   20   0  28588  6872  5464 S  32680  72  72  72  72  72  72  72  72  72
0.3 named     25      7:31.12  1.8   20   0  548272 18872 5584 S   1  25  25  25  25  25  25  25  25  25
0.3 dan      1028    0:00.08  0.6   20   0  152828  6248  2548 S  30580 1828 1828 1828 1828 1828 1828 1828 1828 1828
```

Stratergergy

- 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



A terminal window screenshot showing system statistics and a list of processes. The top part displays memory usage: 0 used, 620984 buff/cache, 615976 avail Mem. Below this is a table of processes with columns for PID, CPU, USER, UID, TIME+, MEM, PR, NI, VIRT, RES, SHR, S, PPID, and others. The processes listed are splunkd, tcpdump, named, vim, and python.

PID	CPU	USER	UID	TIME+	MEM	PR	NI	VIRT	RES	SHR	S	PPID	OTHER
32616	0.3	splunkd	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	named
32695	0.3	splunk	1000	0:02.45	0.7	20	0	28588	6872	5464	S	32680	72 tcpdump
590	0.3	named	72	7:31.12	1.8	20	0	548272	18872	5584	S	1	25 named
1602	0.3	vim	25	0:00.08	0.6	20	0	152828	6248	2548	S	38584	1828 vim
1243	0.3	python	1028										

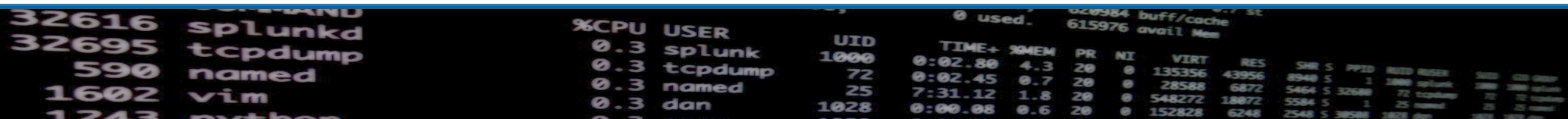
Which systems?

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

		%CPU	USER	UID	TIME+	MEM	PR	NI	VIRT	RES	SHR	S	PPID	PPID	USER	GROUP
32616	splunkd	0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	3000	splunk	splunk
32695	tcpdump	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	72	tcpdump	tcpdump
590	named	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25	named	named
1602	vim	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	30580	1602	vim	vim
1243	python	0.3														

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?)



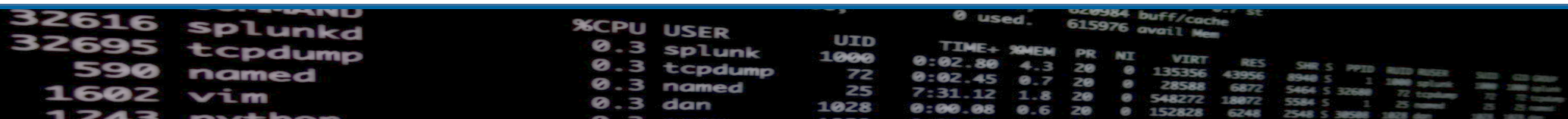
The image shows a terminal window with two distinct sections. The top section displays system logs with entries for 'splunkd', 'tcpdump', 'named', 'vim', and 'python'. The bottom section shows the output of the 'top' command, displaying system statistics and a list of running processes.

System Logs	
32616	splunkd
32695	tcpdump
590	named
1602	vim
1243	python

%CPU	USER	UID	TIME+	PMEM	PR	NI	VIRT	RES
0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956
0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872
0.3	named	25	7:31.12	1.8	20	0	548272	18872
0.3	dan	1028	0:00.08	0.6	20	0	152828	6248

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



A screenshot of a Splunk forwarder's process list, likely from a 'top' or 'ps' command. The table shows various processes running under the 'splunk' user, including 'splunkd', 'tcpdump', 'named', 'vim', and 'python'. The columns include PID, CPU usage, user, UID, TIME+, MEM, PR, NI, VIRT, RES, and SHR. The 'splunkd' process is the most prominent, showing high CPU usage and memory.

PID	%CPU	USER	UID	TIME+	MEM	PR	NI	VIRT	RES	SHR	S	PPID	PPID	NI	NI	SHR	S
32616	0.3	splunkd	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	1000	1000	1000	1000	1000
32695	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	1	1000	1000	1000	1000	1000
590	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	1000	1000	1000	1000	1000
1602	0.3	vim	1028	0:00.08	0.6	20	0	152828	6248	2548	S	1	1000	1000	1000	1000	1000
1243	0.3	python															

Some nice blacklist options to start with

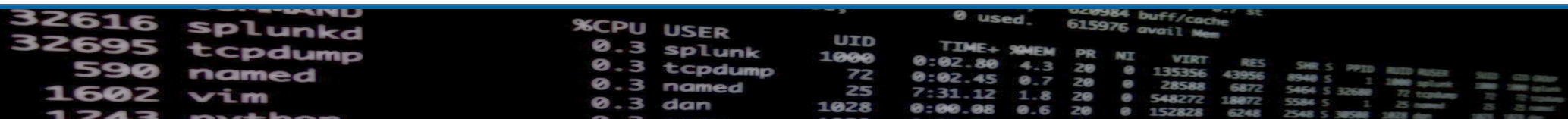
- <https://gist.github.com/automine/a3915d5238e2967c8d44b0ebcfb66147>

```
1 [WinEventLog://Security]
2 disabled = 0
3 start_from = oldest
4 current_only = 0
5 evt_resolve_ad_obj = 1
6 checkpointInterval = 5
7 blacklist1 = EventCode="4662" Message="Object Type:\s+(?!groupPolicyContainer)"
8 blacklist2 = EventCode="566" Message="Object Type:\s+(?!groupPolicyContainer)"
9 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)"
```

		%CPU	USER	UID	TIME+	PMEM	PR	NI	VIRT	RES	SHR	S	PPID	PPID	NAME	GROUP
32616	splunkd	0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	3888	splunk	splunk
32695	tcpdump	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	72	tcpdump	splunk
590	named	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25	named	splunk
1602	vim	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38588	1602	vim	splunk
1243	python	0.3	python	1000	0:00.00	0.0	20	0	152828	6248	2548	S	38588	1243	python	splunk

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

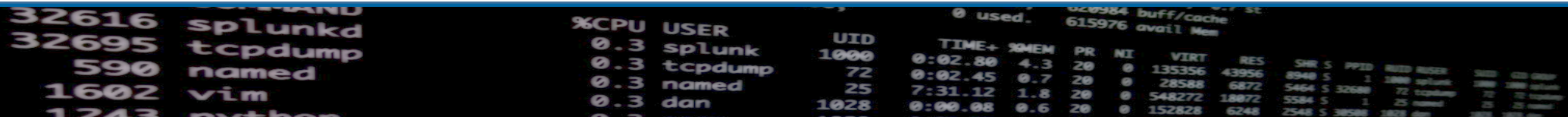


The screenshot shows a terminal window with system metrics and process information. On the left, there's a list of processes with their PIDs and names: 32616 splunkd, 32695 tcpdump, 590 named, 1602 vim, and 1243 python. On the right, there's a table with columns: %CPU, USER, UID, TIME+, %MEM, PR, NI, VIRT, RES, SHR, S, PPID, and PWD. The table lists processes for splunk, tcpdump, named, and dan.

	%CPU	USER	UID	TIME+	%MEM	PR	NI	VIRT	RES	SHR	S	PPID	PWD
32616 splunkd	0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	/usr/sbin/splunkd
32695 tcpdump	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	/usr/sbin/tcpdump
590 named	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	/usr/sbin/named
1602 vim	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	30580	/usr/bin/vim

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



	%CPU	USER	UID	TIME+	MEM	PR	NI	VIRT	RES	SHR	S	PPID	PID	NAME	MEM	MEM	MEM
32616	0.3	splunkd	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	3080	splunkd	1000	1000	splunkd
32695	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	72	tcpdump	72	72	tcpdump
590	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25	named	25	25	named
1602	0.3	vim	1028	0:00.08	0.6	20	0	152828	6248	2548	S	30580	1602	vim	1602	1602	vim
1243	0.3	python															

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

32616	splunkd	%CPU	USER	UID	TIME+	MEM	PR	NI	VIRT	RES	SHR	S	PPID	NAME	STATE	TIME	TIME	TIME
32695	tcpdump	0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	2880	python	2880	2880	python
590	named	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	72	tcpdump	72	72	tcpdump
1602	vim	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25	named	25	25	named
1243	python	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38580	1602	vim	1602	1602	vim

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

	%CPU	USER	UID	TIME+	MEM	PR	NI	VIRT	RES	SHR	S	PPID	RUDD	MUSER	STAT	CSD	GROUP
32616 splunkd	0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	1000	splunk	1000	1000	splunk
32695 tcpdump	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	72	tcpdump	72	72	tcpdump
590 named	0.3	named	25	7:31.12	1.8	20	0	548272	18072	5584	S	1	25	named	25	25	named
1602 vim	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	30500	1028	dan	1028	1028	dan

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

```
020984 buff/cache 0 used, 615976 avail Mem
32616 splunkd
32695 tcpdump
590 named
1602 vim
1243 python

%CPU USER      UID      TIME+  %MEM  PR  NI  VIRT  RES  SHR  S  PPID  RSS  RSIZE  GROUP
0.3 splunk  1000  0:02.80  4.3  20  0  135356 43956 8940 S  1  2880 splunk  2880  420 group
0.3 tcpdump  72  0:02.45  0.7  20  0  28588  6872  5464 S  32680  72  tcpdump  2880  2880 splunk
0.3 named    25  7:31.12  1.8  20  0  548272 18872  5584 S  1  25 named  25  25 named
0.3 dan      1028  0:00.08  0.6  20  0  152828  6248  2548 S  38584 1828  dan  25  25 dan
```

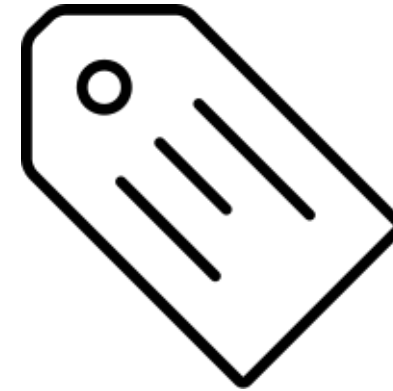
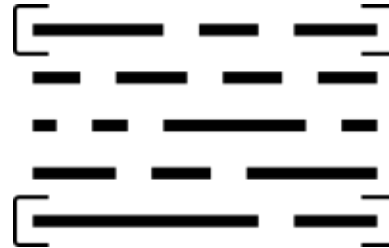
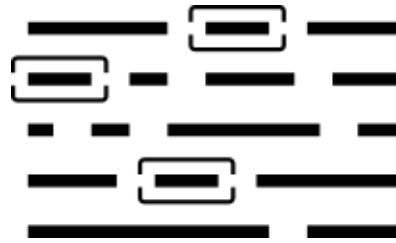
Filtering and cleaning up

- 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)

```
1 [WinEventLog:Security]
2 #Returns most of the space savings XML would provide
3 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
4 SEDCMD-clean1-summary = s/This event is generated[\S\s\r\n]+$/g
5 SEDCMD-clean2-cert_summary = s/Certificate information is only[\S\s\r\n]+$/g
6 SEDCMD-clean3-blank_ipv6 = s/::ffff://g
7 SEDCMD-clean4-token_elevation_summary = s/Token Elevation Type indicates[\S\s\r\n]+$/g
8 SEDCMD-clean5-firewall_summary = s/(?ms)(The Windows Filtering Platform has permitted.*$)//g
9 SEDCMD-clean6-network_share_summary = s/(?ms)(A network share object was checked to see whether.*$)//g
10 SEDCMD-clean7-authentication_summary = s/(?ms)(The computer attempted to validate the credentials.*$)//g
11 SEDCMD-clean8-local_ipv6 = s/(?ms)(::1)//g
```

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

		%CPU	USER	UID	TIME+	PMEM	PR	NI	VIRT	RES	SHR	S	PPID	PPID	NAME	GROUP
32616	splunkd	0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	3880	splunk	splunk
32695	tcpdump	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	72	tcpdump	tcpdump
590	named	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25	named	named
1602	vim	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38580	1602	vim	vim
1243	python	0.3														



Making Them More Useful

		%CPU	USER	UID	TIME+	MEM	PR	NI	VIRT	RES	SHR	S	PPID	PPID	USER	GROUP
32616	splunkd	0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	2880	splunk	2880
32695	tcpdump	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	72	tcpdump	2880
590	named	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25	named	25
1602	vim	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38584	1602	dan	25
1243	python	0.3														

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](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](https://bitbucket.org/SPLServices/seckit_ta_microsoft_windows) (for use with SecKit)

[illegible]