

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

David Shpritz

Aplura, LLC

Baltimore Area Splunk User Group June, 2017

A terminal window showing system statistics and a list of processes. The top part shows memory usage: 0 used, 615976 avail Mem. Below that is a table of processes with columns for PID, COMMAND, %CPU, USER, UID, TIME+, %MEM, PR, NI, VIRT, RES, SHR, S, PPID, PWD, USER, and COMMAND.

PID	COMMAND	%CPU	USER	UID	TIME+	%MEM	PR	NI	VIRT	RES	SHR	S	PPID	PWD	USER	COMMAND
32616	splunkd	0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8948	S	1	/usr	splunk	splunkd
32695	tcpdump	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32689	/usr	tcpdump	tcpdump
590	named	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	/usr	named	named
1602	vim	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38588	/usr	dan	vim

Agenda

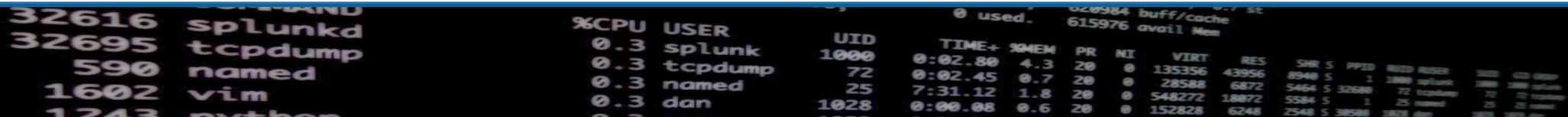
- Getting Windows Events into Splunk: Patterns and Practices
- TURN DOWN THE VOLUME: License reduction tips
- Making them more useful: Improving knowledge objects

```
0 used, 615976 avail Mem
```

	%CPU	USER	UID	TIME+	MEM	PR	NI	VIRT	RES	SHR	S	PPID	PPID	USER	MEM	MEM	MEM
32616	0.3	splunkd	1000	0:02.80	4.3	20	0	135356	43956	8948	S	1	2888	splunk	2888	2888	splunk
32695	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32688	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	38588	1602	vim	1602	1602	vim
1243		python															

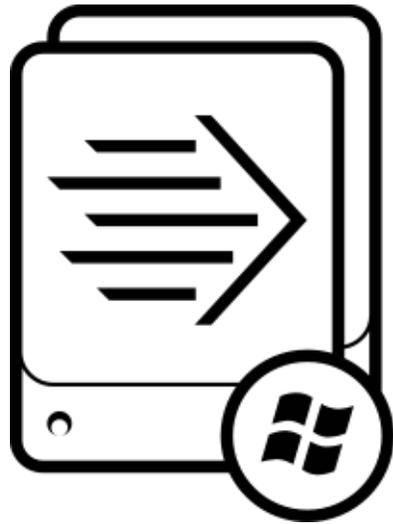
Ground Rules

- 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



The image shows a terminal window with two sections of output. The top section displays system statistics, including memory usage (0 used, 615976 avail Mem) and a list of processes with their PIDs and names. The bottom section shows a detailed process list with columns for %CPU, USER, UID, TIME+, %MEM, PR, NI, VIRT, RES, SHR, S, PPID, and other fields.

PID	NAME	%CPU	USER	UID	TIME+	%MEM	PR	NI	VIRT	RES	SHR	S	PPID
32616	splunkd	0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8948	S	1
32695	tcpdump	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32689
590	named	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1
1602	vim	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38588



Getting Windows Events into Splunk

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

%CPU USER      UID      TIME+  %MEM  PR  NI  VIRT  RES  SHR  S  PPID  RSS  RSIZE  STATE  CWD  OOM
0.3 splunk  1000    0:02.80  4.3  20  0  135356 43956 8948 S  1  2888 /opt/splunk 2888 /opt/splunk
0.3 tcpdump  72     0:02.45  0.7  20  0  28588  6872  5464 S  32688  72 /opt/splunk 72 /opt/splunk
0.3 named    25     7:31.12  1.8  20  0  548272 18872  5584 S  1  25 /usr/sbin 25 /usr/sbin
0.3 dan     1028    0:00.08  0.6  20  0  152828  6248  2548 S  38588 1828 /usr/bin 38588 /usr/bin
```


Windows Event Forwarding

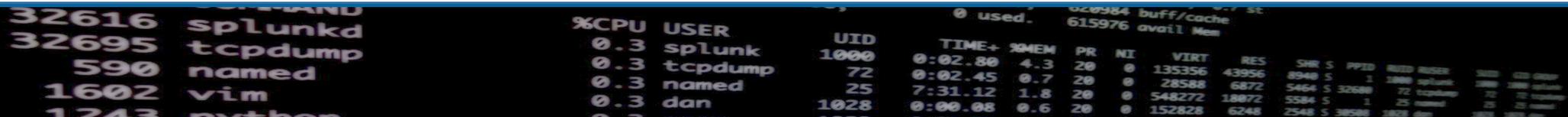
- Native to Windows (2008R2 and up)
- Pros
 - Native to Windows, no agent
 - Can be configured with GPO
- Cons
 - Almost high fedlity
 - Slower
 - Customer testing shows it consumes more resources than a UF

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

%CPU USER      UID      TIME+  %MEM  PR  NI  VIRT  RES  SHR  S  PPID  RSS  RSD  RSDR  ST
0.3 splunk  1000    0:02.80  4.3  20  0  135356  43956  8940  S  1  2880  1000  1000  0000  0000
0.3 tcpdump  72      0:02.45  0.7  20  0  28588  6872  5464  S  32680  72  10000  1000  1000  0000
0.3 named    25      7:31.12  1.8  20  0  548272  18872  5584  S  1  25  10000  1000  1000  0000
0.3 dan      1028    0:00.08  0.6  20  0  152828  6248  2548  S  38580  1028  1000  1000  1000  0000
```

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 of your Windows hosts)
 - Low fidelity
 - Dealing with permissions

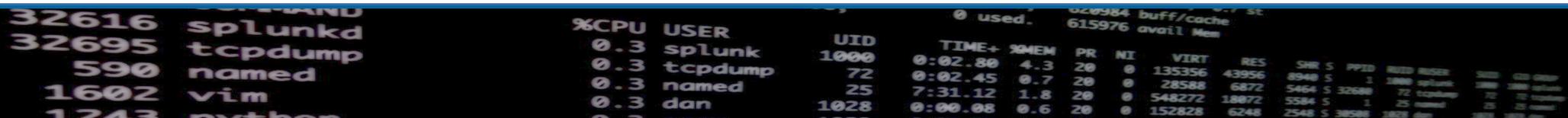


A terminal window showing system metrics and a process list. The top part shows memory usage: 0 used, 620984 buff/cache, 615976 avail Mem. Below that is a table with columns: %CPU, USER, UID, TIME+, %MEM, PR, NI, VIRT, RES, SHR, S, PPID, RSS, RSIZE, RTEXT, RDATA, RSTACK, RCODE, RLIB, RSHR, RTEXT, RDATA,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

%CPU	USER	UID	TIME+	%MEM	PR	NI	VIRT	RES	SHR	S	PPID	RSS	RSIZE	RTEXT	RDATA	RSTACK	RCODE	RLIB	RSHR	RTEXT	RDATA	RSTACK	RCODE	RLIB	RSHR
0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8948	S	1	2880	splunk	2880	2880	splunk									
0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	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									
0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38588	1028	dan	1028	1028	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 showing system metrics and a process list. The top part shows memory usage: 0 used, 620984 buff/cache, 615976 avail Mem. Below that is a table of processes with columns for PID, CPU, USER, UID, TIME+, MEM, PR, NI, VIRT, RES, SHR, S, PPID, and other system details.

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	8948	S	1	2888
32695	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32688	72
590	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25
1602	0.3	vim	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38588	1602
1243	0.3	python											

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?

```
020984 buff/cache 0 used, 615976 avail Mem
```

PPID	USER	%CPU	MEM	TIME+	PR	NI	VIRT	RES	SHR	S	PPID	USER	%CPU	MEM	TIME+	PR	NI	VIRT	RES	SHR	S	
32616	splunkd	0.3		0:02.80	20	0	135356	43956	8948	S	1	32695	tcpdump	0.3	0:02.45	20	0	28588	6872	5464	S	1
32695	tcpdump	0.3		7:31.12	20	0	548272	18872	5584	S	1	1602	vim	0.3	0:00.08	20	0	152828	6248	2548	S	1
590	named	0.3										1243	python									
1602	vim	0.3																				
1243	python	0.3																				



TURN DOWN THE VOLUME: License reduction tips

```
0 used, 615976 avail Mem
```

PPID	COMMAND	%CPU	USER	UID	TIME+	PMEM	PR	NI	VIRT	RES	SHR	S	PPID	PPID	PMEM	PR	NI	VIRT	RES
32616	splunkd	0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8948	S	1	3888	splunk	2000	2000	13880	420
32695	tcpdump	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32688	72	tcpdump	72	72	13880	420
590	named	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25	named	25	25	13880	420
1602	vim	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38588	1602	vim	1602	1602	13880	420

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?



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

%CPU USER      UID      TIME+  %MEM  PR  NI  VIRT  RES  SHR  S  PPID  RSS  RSIZE  CHILD  STATE
0.3 splunk  1000    0:02.80  4.3  20  0  135356  43956  8948  S  1  3888  3888  3888  3888  3888  3888  3888  3888
0.3 tcpdump  72     0:02.45  0.7  20  0  28588  6872  5464  S  1  72  1024  72  1024  72  1024  72  1024
0.3 named    25     7:31.12  1.8  20  0  548272  18872  5584  S  1  25  3888  25  3888  25  3888  25  3888
0.3 dan     1028   0:00.08  0.6  20  0  152828  6248  2548  S  1  3888  3888  3888  3888  3888  3888  3888  3888
```

Stratergery

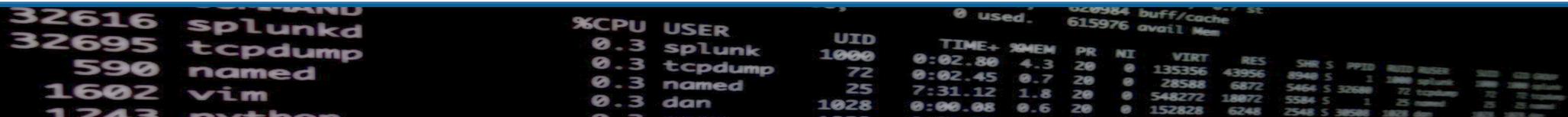
- Pick your systems carefully
- Pick your inputs carefully on those systems
- Whitelist and Blacklist carefully
- Resolving objects
- Baseline?
- Current_only? Start_from?
- XmlWinEventLog
- Filtering and cleaning up

```
020984 buff/cache 0 used, 615976 avail Mem
```

	%CPU	USER	UID	TIME+	PMEM	PR	NI	VIRT	RES	SHR	S	PPID	RUDD	RUSER	SSID	CSID	GROUP
32616	0.3	splunkd	1000	0:02.80	4.3	20	0	135356	43956	8948	S	1	2888	splunk	2888	2888	splunk
32695	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32688	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	38588	1602	vim	1602	1602	vim
1243	0.3	python															

Picking your inputs (not your nose)

- Set a baseline for which logs ALL of 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 of 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?)



A terminal screenshot showing system metrics and process information. The top part shows memory usage: 0 used, 620984 buff/cache, 615976 avail Mem. Below that is a table with columns: %CPU, USER, UID, TIME+, %MEM, PR, NI, VIRT, RES. The table lists processes like splunk, tcpdump, named, vim, dan, and others with their respective resource usage.

%CPU	USER	UID	TIME+	%MEM	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

```
020984 buff/cache 0 used, 615976 avail Mem
```

PPID	PID	USER	%CPU	MEM	TIME+	PR	NI	VIRT	RES	SHR	S	PPID	PPID	USER	%CPU	MEM	TIME+	PR	NI	VIRT	RES	SHR	S
32616	32616	splunkd	0.3	4.3	0:02.80	20	0	135356	43956	8948	S	1	32616	splunkd	0.3	4.3	0:02.80	20	0	135356	43956	8948	S
32695	32695	tcpdump	0.3	0.7	0:02.45	20	0	28588	6872	5464	S	32695	32695	tcpdump	0.3	0.7	0:02.45	20	0	28588	6872	5464	S
590	590	named	0.3	1.8	7:31.12	20	0	548272	18872	5584	S	1	590	named	0.3	1.8	7:31.12	20	0	548272	18872	5584	S
1602	1602	vim	0.3	0.6	0:00.08	20	0	152828	6248	2548	S	32616	1602	vim	0.3	0.6	0:00.08	20	0	152828	6248	2548	S
1243	1243	python	0.3											python	0.3								

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+	MEM	PR	NI	VIRT	RES	SHR	S	PPID	PPID	USER	MEM	MEM	MEM
32616	0.3	splunkd	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	3888	splunk	3888	3888	splunk
32695	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32688	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	38588	1602	vim	1602	1602	vim
1243	0.3	python															

Current_only vs. start_from

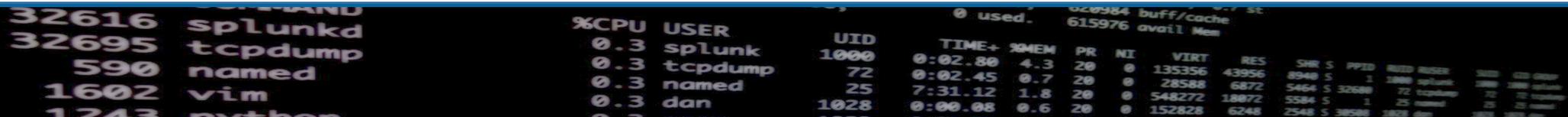
- Current_only tells Splunk to only grab the latest events (like tail -f, if Windows had such a thing)
- Useful to make sure you don't get all of the historical data
- May want to set that to "true" on initial deployment
- Then set to "false", restart, and it should pick up from the checkpoint
- Start_from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

A terminal window showing system statistics and a list of processes. The top part shows system statistics like '0 used' and '615976 avail Mem'. Below that is a table of processes with columns for PID, USER, %CPU, and UID. The processes listed are splunkd, tcpdump, named, vim, and python.

PID	USER	%CPU	UID
32616	splunkd	0.3	1000
32695	tcpdump	0.3	72
590	named	0.3	25
1602	vim	0.3	1028
1243	python	0.3	

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



A terminal window showing system resource usage. The left side lists processes with their PIDs and names. The right side shows a detailed table of resource usage for the 'splunk' process.

COMMAND	%CPU	USER	UID	TIME+	PMEM	PR	NI	VIRT	RES	SHR	S	PPID	PPID	PPID	PPID	PPID	PPID	PPID	PPID
32616 splunkd	0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8948	S	1	1000	1000	1000	1000	1000	1000	1000
32695 tcpdump	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32689	72	1000	1000	1000	1000	1000	1000
590 named	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25	1000	1000	1000	1000	1000	1000
1602 vim	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38588	1028	1000	1000	1000	1000	1000	1000
1243 other	0.3																		

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

```
0 used, 615976 avail Mem
```

PPID	PID	%CPU	USER	UID	TIME+	MEM	PR	NI	VIRT	RES	SHR	S	PPID	RUSER	RUID	ST	TTY	TIME	COMMAND
32616	32616	0.3	splunkd	1000	0:02.80	4.3	20	0	135356	43956	8948	S	1	32616	splunkd	1000	0	0:02.80	splunkd
32695	32695	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32688	72	tcpdump	72	0	0:02.45	tcpdump
590	590	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25	named	25	0	7:31.12	named
1602	1602	0.3	vim	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38588	1602	vim	1602	0	0:00.08	vim
1243	1243	0.3	python	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38588	1243	python	1243	0	0:00.08	python

Filtering and cleaning up

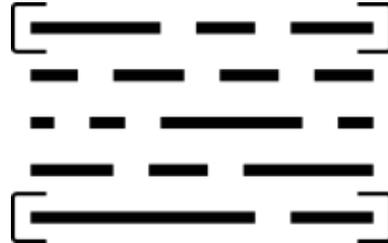
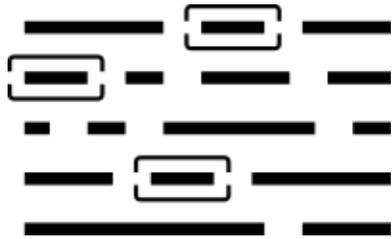
- IPv6 support in event logs results in a lot of “::” and “ffff” and other garbage
- Let’s clean up a lot (thanks to a lot of people for this)

```
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>

PID	Process Name
32616	splunkd
32695	tcpdump
590	named
1602	vim
1243	python

%CPU	USER	UID	TIME+	MEM	PR	NI	VIRT	RES	SHR	S	PPID	PPID	PPID	PPID	PPID	PPID	PPID	PPID	PPID
0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	3888	aplunk	3888	3888	3888	3888	3888	3888
0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	72	tcpdump	72	72	72	72	72	72
0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25	named	25	25	25	25	25	25
0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38588	38588	38588	38588	38588	38588	38588	38588	38588



Making Them More Useful

```
0 used, 620984 buff/cache, 615976 avail Mem
%CPU USER      UID      TIME+  %MEM  PR  NI  VIRT  RES  SHR  S  PPID  RSS  RSIZE  CHILD  GROUP
32616 splunkd    1000     0:02.80  4.3  20  0  135356 43956 8940 S  1  2880 splunkd  2880  2880 splunkd
32695 tcpdump    72      0:02.45  0.7  20  0  28588  6872  5464 S  1  72 tcpdump  72  72 tcpdump
590  named      25      7:31.12  1.8  20  0  548272 18872 5584 S  1  25 named    25  25 named
1602 vim        1028    0:00.08  0.6  20  0  152828  6248  2548 S  1  2528 vim     2528  2528 vim
1243 python
```

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)

A terminal window showing system statistics and a list of processes. The top part shows memory usage: 0 used, 615976 avail Mem. Below that is a table of processes with columns for PID, COMMAND, %CPU, USER, UID, TIME+, MEM, PR, NI, VIRT, RES, SHR, S, PPID, and other details.

PID	COMMAND	%CPU	USER	UID	TIME+	MEM	PR	NI	VIRT	RES	SHR	S	PPID	OTHER
32616	splunkd	0.3	splunk	1000	0:02.80	4.3	20	0	135356	43956	8940	S	1	2880 splunk
32695	tcpdump	0.3	tcpdump	72	0:02.45	0.7	20	0	28588	6872	5464	S	32680	72 tcpdump
590	named	0.3	named	25	7:31.12	1.8	20	0	548272	18872	5584	S	1	25 named
1602	vim	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38588	1602 vim
1243	python	0.3	dan	1028	0:00.08	0.6	20	0	152828	6248	2548	S	38588	1243 python