FIRST CHANGE PASSWORD:

Passwd. Then passwd useryouwanttochange.

For web ui user: /opt/splunk/bin/splunk edit user admin -password ‘putstrongpasswordhere’ -role admin -auth admin:changeme

IF NO GUI, DO THIS:

Sudo dnf update -y

Dnf repolist all (check if codeready\_builder is enabled or disabled)

Sudo dnf config-manager –-enable ol9\_codeready\_builder (that an o not zero and l not one)

sudo dnf install epel-release -y

sudo dnf install kde-plasma-desktop

Verify with repolist all again

sudo dnf -y upgrade

sudo dnf groupinstall -y “KDE Plasma Workspaces”

ln -sf /lib/systemd/system/runlevel5.target /etc/systemd/system/default.target

systemctl set-default graphical.target

sudo reboot

Login into your sysadmin (or whatever admin account) then open up a terminal

Cd /opt/splunk #change directory first

./bin/splunk start # start the ports to accept incoming data

./bin/splunk status # check if it’s working

./bin/splunk restart # use this if you make config file changes

NOW WE NEED TO ALLOW THE PORTS

sudo systemctl status firewalld   
# If not running, start it

sudo systemctl start firewalld

sudo systemctl enable firewalld

# Open the port

sudo firewall-cmd --permanent --add-port=9997/tcp

sudo firewall-cmd --permanent --add-port=8089/tcp

sudo firewall-cmd --permanent --add-port=8000/tcp

sudo firewall-cmd --permanent --add-port=8191/tcp

sudo firewall-cmd --permanent --add-port=9887/tcp

sudo firewall-cmd --permanent --add-port=8080/tcp

# Reload firewall to apply changes

sudo firewall-cmd –reload

# Verify the port is open

sudo firewall-cmd --list-ports

# Test port connectivity

nc -zv localhost 8000 # then 8089 and 9997

FORWARDERS-------------------------------------------

Check for the version of splunk you’re using and get the matching forwarder version. If you can’t access splunk website to get the download, use a real machine to copy the wget link. (I know it sucks but it works)… OH and pay attention if it’s windows or Linux.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

THIS PART IF SPLUNK FORWARDER IS ON LINUX.

Locate Correct splunk version and download the correct package version (or copy wget into terminal)

Mkdir ~/Downloads/www # create the directory to put the forwarder

mv splunkforwarder ~/Downloads/www 🡪 cd /Downloads/www 🡪 ./bin/splunk start –accept-license 🡪 press y then change port to 9997 when asked.

install Firefox or chromium then into browser // localhost 8000

Login 🡪 settings in top tab 🡪 forwarding and receiving 🡪 configure receiving 🡪 new receiving port 🡪 port 9997 and save.

Cd /Downloads/www/splunkforwarder/bin

./splunk add forward-server (Splunk machine IP):9997 #don’t include () with your Ip. AND THE IP IS THE SPLUNK MACHINE, NOT SOMEONE ELSES IP THEY DOWNLOADED IT TOO.

next, we will tell Splunk forwarder which logs files to monitor.

./splunk add monitor /var/log -index Linux\_host

# we could specify /var/log/auth.log or /syslog

Then lets confirm it works

Cd /Downloads/www/splunkforwarder/etc/apps/search/local

ls # should see inputs.conf 🡪 cat inputs.conf

A screenshot of a computer error

AI-generated content may be incorrect.

if disabled = true or 1. do this 🡪 nano into inputs.conf and change the value to false or 0 🡪 sudo

Your outputs.conf should look like this. If not, build it yourself using text editor.

A screenshot of a computer

AI-generated content may be incorrect.

/Downloads/www/splunkforwarder/bin/splunk restart

and then sudo /opt/splunk/bin/splunk restart

create a test log to check if things work

cd /Downloads/www/splunkforwarder/bin/ 🡪 logger “put-whatever-you-want-with-dashes” 🡪 tail -1 /var/log #or what specific logs you want 🡪 use splunk Web UI 🡪 index=Linux\_host 🡪 then you should see the logger message

THIS PART IF SPLUNK IS ON WINDOWS.

Open Windows Defender Firewall with Advanced Security 🡪Select "Inbound Rules" and click "New Rule..." in the Actions panel 🡪Select "Port" and click "Next" 🡪Select "TCP" and enter the specific port(s) separated by commas (8000,8089,9997,8191)🡪Select "Allow the connection" and click "Next"🡪Select which network types apply (Domain, Private, Public) and click "Next" 🡪Name the rule "Splunk Ports" with description and click "Finish"

Locate Correct splunk version and download the correct package version. (or copy wget into terminal) IF THE SPLUNK FORWARDER HAS TO BE 9.1.1, CHECK WGET LINK AND AVOID THE ONE WITH AIRGAP A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

Description automatically generated

User a secure password.

If it asks for other stuff like ssl, just skip over it or accept its default option.

A screenshot of a computer

Description automatically generated check the last box if its an AD machine.

A red and white box with black text

Description automatically generatedreplace IP with the Splunk IP and port as 8089.

Leave indexer blank

IF this machine is the indexer-------------

Web UI 🡪 settings 🡪 forwarding management 🡪 should show the IPs that have configured the forwarding to you properly.

Back to settings 🡪 add data 🡪 forward 🡪 select hosts and name it whatever (I like to name it the machine version and the critical service like: AD2019\_HTTP)

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Review then submit. Happy hunting

Before searching through logs, view search history and data summary to get a feel of what you’re dealing with. (If there even is a history)

A screenshot of a computer

Description automatically generated

In the Add Data tab, Monitor option is used to collect data from files and ports

List of filters (if you want excluding, use !=)

Source\_ip=””

Source\_Country=””

A screenshot of a computer program

Description automatically generated

Use sourcetype in front. EX: sourcetype=stream:http

Here’s an example scenario with imreallynotbatman.com as the enemy:

***What is the web scanner, the attacker used to perform the scanning attempts?***

I used the following query:  
index=botsv1 imreallynotbatman.com sourcetype=Suricata eventtype=suricata\_eve\_ids\_attack dest="imreallynotbatman.com" dest\_ip="192.168.250.70" status=404.

Some filters may not be necessary, but my goal was to narrow down the search results as much as possible. Here’s my thought process:

* The eventtype filter was used because we are under attack, as detected by the IDS.
* The dest and dest\_ip filters were included because the destination indicates we are being scanned.
* The status=404 filter was added because scanning often results in 404 errors when web scanners check for non-existent subdirectories.

sourcetype=stream\* | stats count(src\_ip) as Requests by src\_ip | sort – Requests (include whatever index it uses first.)

This query uses the stats function to display the count of the IP addresses in the field src\_ip.

|table src\_ip user  
|dedup user

This helps filter for users.

Remember you can use =, !=, AND, OR, NOT in searches.

EX: index=windowslogs AccountName !=SYSTEM AND AccountName=James

Don’t want the system user to clutter the search want to see what James is doing.

A screenshot of a computer

Description automatically generated

EX: term cyber returns 0, but cyber\* return 12256.

Filtering with SPL:

Fields command is used to add or remove mentioned fields from the search results. To remove the field, minus sign ( - ) is used before the fieldname and plus ( + ) is used before the fields which we want to display.

| fields + HostName – EventID

index=windowslogs | fields + host + User + SourceIp

Dedup is the command used to remove duplicate fields from the search results. We often get the results with various fields getting the same results. These commands remove the duplicates to show unique values.

index=windowslogs | table EventID User Image Hostname | dedup EventID

index=windowslogs | table \_time EventID Hostname SourceName | dedup Hostname | reverse

Remember you can use head tail, sort. (Head and tail default to top 10/bottom 10, so specify number if you want more or less.)

| head 5

| tail 20

Top: This command returns frequent values for the top 10 events.

| top limit=8 Image

rare: This command does the opposite of top command as it returns the least frequent values or bottom 10 results.

Highlight: The highlight command shows the results in raw events mode with fields highlighted.

index=windowslogs | highlight User, host, EventID, Image

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

For visualization filter: use chart count by or timechart count by.

Ex: index=windowslogs | chart count by Image

Example uses to filters:

the big 4 to look for:

id.orig\_h

id.orig\_p

id.resp\_h

id.resp\_p

\*REPLACE Corelight\_conn with the actual sourcetype aka the machine you’re viewing\*

local\_orig: false -external connection (if true, that's internal)

Conn.log

\*sourcetype=corelight\_conn (since I'll be using the open source it'll just be sourcetype=conn)

\*sourcetype=corelight\_conn \*\*local\_orig=true\*\* local\_resp=true (conn\_state=S0 OR conn\_state=REJ) //simple threat hunt for TCP scans specifically for internal traffic, switch the local\_orig=false if you want external

sourcetype=corelight\_conn local\_orig=true local\_resp=true (conn\_state=S0 OR conn\_state=REJ) | \*\*stats count by id.orig\_h | sort -count\*\* //group on source IPs

sourcetype=corelight\_conn local\_orig=true local\_resp=true (conn\_state=S0 OR conn\_state=REJ) | \*\*stats dc(id\_resp\_p) by id.orig\_h\*\* | sort -count //count number of destination ports, could also add dc(id\_resp\_h)

sourcetype=corelight\_conn local\_orig=true local\_resp=true (conn\_state=S0 OR conn\_state=REJ) | \*\*stats dc(id.orig\_h) dc(id.resp\_h) by sourcetype\*\* //refining TCP scan hunt to count multiple things

sourcetype=corelight\_conn local\_orig=true local\_resp=true (conn\_state=S0 OR conn\_state=REJ) | stats dc(id.orig\_h) dc(id.resp\_h) \*\*count\*\* by sourcetype //TCP scan hunt to add a raw count

sourcetype=corelight\_conn local\_orig=true local\_resp=true (conn\_state=S0 OR conn\_state=REJ) | \*\*stats dc(id.orig\_h) as "# sources" by id.resp\_h\*\* //make it more readable

sourcetype=corelight\_conn local\_orig=true local\_resp=true (conn\_state=S0 OR conn\_state=REJ) | \*\*stats dc(id.resp\_h) as "#\_dports" dc(id.resp\_p) as "#\_dst" by id.orig\_h\*\* // this counts ports.. IF THE \_DPORTS NUMBER IS HIGH, THEY ARE SCANNING VERTICALLY, IF \_DST NUMBER IS HIGH, THEY ARE SCANNING HORIZONTALLY, BOTH ARE MOST LIKELY BAD. If the IP isn't familiar, John Wick it.

SSH Log

sourcetype=corelight\_conn service=ssh local\_orig=false local\_resp=true // Check if you have SSH activity(if your not suppose to have ssh, this will show if there is)

sourcetype=corelight\_ssh | table id.orig\_h, client, id.resp\_p, server, auth\_success // Look for strange SSH behaviors, including on non-standard ports. \*\*Attackers can hide their SSH activity by running it on 80 or 443 ports, but you can find SSH traffic where the destination port is not 22.\*\*

sourcetype=corelight\_ssh id.orig\_h=44.201.4.198 id.resp\_h=10.0.35.11 | table ts, client, server, id.resp\_h, id.resp\_p, auth\_attempts, auth\_success | sort ts // detailed report on did this IP address make a successful connection, did the IP get what it wanted. (could also replace server with hassh) (could also add to the table such as, service, conn\_state, history, proto)

HTTP Log

sourcetype=corelight\_http status\_code >=400 // \*\*searches for transactions that are rejected or cause server errors.\*\*

sourcetype=corelight\_http status\_code >=400 | \*\*stats count by id.orig\_h\*\* // group ups the IPs

sourcetype=corelight\_http status\_code >=400 | \*\*stats dc(uri) as "#\_uris"\*\* count by id.orig\_h // unique URIs the client touched.

sourcetype=corelight\_http | \*\*mvexpand resp\_mime\_types{} | stats count by resp\_mime\_types{}\*\* // find types of files downloaded over HTTP

sourcetype=corelight\_http | mvexpand resp\_mime\_types{} | search resp\_mime\_types{}="\*exec\*" | stats count by resp\_mime\_types{} uri // refine to focus on only executables (in this , there were .png files that aren't actually png files and woff is bad too.)

SSL and X509 Logs

sourcetype=corelight\_x509 | stats dc(certificate.subject) by certificate.issuer // sort by rare certificate issuers

certificates that don't include a canonical name (CN) must be investigated.

after investigating, you'd use sourcetype=ssl and filter by the name you found

sourcetype=corelight\_ssl issuer="C=US"

DNS Logs (most valuable)

sourcetype=corelight\_dns | stats dc(icann\_host\_subdomain) by icann\_domain // finding which domain has largest number of subdomains

sourcetype=corelight\_dns | stats dc(icann\_host\_subdomain), \*\*dc(id.orig\_h)\*\* by icann\_domain // adjusting to add unique number of clients

sourcetype=corelight\_dns | stats dc(query) as "query\_counts" by id.orig\_h // counts number of unique queries.

if your query has a bunch of characters, your data is being exfiltrated.

if you see a high number of queries but the query name is normal, someone is continuously trying to enumerate.