corelight

Tech brief

Splunk Enterprise Security DNS correlation

Background

This brief provides technical background, configuration specifics, and usage guidance supporting the blog ***Overwatch on DNS Traffic with Corelight***. The information provided within this brief is intended to help Corelight customers (who are also Splunk Enterprise Security users) properly setup Splunk Enterprise Security data models and correlation searches to realize the full benefits highlighted in the blog post.

Caveats

Every Splunk implementation is different and in some cases details within this brief may not be accurate for every Splunk ES user. This brief assumes the correlation searches were not altered prior to these recommendations. Other assumptions:

- Latest Corelight Sensor code
- Latest Splunk Corelight TA installed
- Splunk version 8.0.1 or newer
- Splunk ES version 6.1.0 or newer
- Splunk ES Content Updates version 1.0.50 or newer
- Splunk CIM version 4.14 or newer
- Corelight data index whitelisted for appropriate data model(s)
- Lookup files in Correlation searches are populated by normal means and with appropriate data

General information

It is important to note that a typical DNS correlation searches assume the data was provided by a DNS log. DNS logs from DNS servers do not combine the request and response so those appear as individual logs. A Corelight DNS log is derived from packet data. The request and response are combined into a single log providing great value. The combined log simplifies many of the correlation searches in Splunk ES.

- DNS.message_type can be removed as a filter from any DNS correlation search
  - This value doesn't natively exist in ANY DNS log (MS, Bind, infoblox)
  - This value would need to be extract by an eval statement in other data sets
  - Not required for Corelight data since all logs are both query and response
  - If you prefer substitutions for DNS.message_type, these are valid for Corelight data (without using an eval statement to extract)
    - DNS.message_type="query" can be replaced by "DNS.query"="*.*"
    - DNS.message_type="response" can be replaced by "DNS.answer"="*.*"

- The following fields should be added to the Network Resolution Data model to significantly reduce correlation search complexity. These fields exist as part of Corelight DNS data and do not need to be calculated with an "eval" statement.
  - answer_length
  - query_length
  - answer_count
  - dns_any (this value is calculated by an eval statement added to the Corelight TA. ***EVAL-dns_any = if((record_type="*"),"true","false")***

- The following fields should be added to the Network Traffic Data model. Some of these will be used to support DNS correlations detailed in this document while others will provide valuable pivot fields for future use cases.
  - service
  - conn_state
  - history
  - fuid
  - community_id

Correlations search info

The correlation searches detailed in this section can be successfully enabled with only Corelight DNS logs. In some instances, the search required no modification. In other cases, searches have been simplified and optimized.

Search text in **bold red** – text was changed for removed

App: DA-ESS-NetworkProtection
[Excessive DNS Failures]

Default search:
| tstats summariesonly=true count from datamodel="Network_Resolution"."DNS" where "DNS.reply_code"!="No error" AND "DNS.reply_code"!="NoError" by "DNS.src", | rename "DNS.src" as "src" | where 'count'>100

Update - No modification required

[Excessive DNS Queries]

Default search:

| tstats summariesonly=true count from datamodel="Network_Resolution"."DNS" **where "DNS.message_type"="QUERY"** by "DNS.src" | rename "DNS.src" as "src" | where 'count'>100

Update – ***removed message_type=query because all Corelight DNS logs contain a query***

| tstats summariesonly=true count from datamodel="Network_Resolution"."DNS" by "DNS.src" | rename "DNS.src" as "src" | where 'count'>100

App: DA-ESS-ContentUpdate

[Clients Connecting to Multiple DNS Servers]

Default search:

| tstats `security_content_summariesonly` count, values(DNS.dest) AS dest dc(DNS.dest) as dest_count from datamodel=Network_Resolution **where DNS.message_type=QUERY** by DNS.src | `drop_dm_object_name("Network_Resolution")` |where dest_count > 5 | `clients_connecting_to_multiple_dns_servers_output_filter`

Update - ***removed message_type=query because all Corelight DNS logs contain a query***

| tstats `security_content_summariesonly` count, values(DNS.dest) AS dest dc(DNS.dest) as dest_count from datamodel=Network_Resolution by DNS.src | `drop_dm_object_name("Network_Resolution")` |where dest_count > 5 | `clients_connecting_to_multiple_dns_servers_output_filter`

[Detect DNS requests to Phishing Sites leveraging EvilGinx2]

Default search:

| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as lastTime values(DNS.answer) as answer from datamodel=Network_Resolution.DNS by DNS.dest DNS.src DNS.query host | `drop_dm_object_name(DNS)`| rex field=query ".*?(?<domain>[^./:]+\.(\S{2,3}|\S{2,3}.\S{2,3}))$" | stats count values(query) as query by domain dest src answer| search `evilginx_phishlets_amazon` OR `evilginx_phishlets_facebook` OR `evilginx_phishlets_github` OR `evilginx_phishlets_0365` OR `evilginx_phishlets_outlook` OR `evilginx_phishlets_aws` OR `evilginx_phishlets_google` | search NOT [inputlookup legit_domains.csv | fields domain]| join domain type=outer [| tstats count `security_content_summariesonly` values(Web.url) as url from datamodel=Web.Web by Web.dest Web.site | rename "Web.*" as * | rex field=site ".*?(?<domain>[^./:]+\.(\S{2,3}|\S{2,3}.\S{2,3}))$" | table dest domain url] | table count src dest query answer domain url

Update – no modification required

[Detect Long DNS TXT Record Response]

Default search:

| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as lastTime from datamodel=Network_Resolution where DNS.message_type=response AND DNS.record_type=TXT by DNS.src DNS.dest DNS.answer DNS.record_type | `drop_dm_object_name("DNS")` **| eval anslen=len(answer) | search anslen>100** |

`security_content_ctime(firstTime)` | `security_content_ctime(lastTime)` | rename src as "Source IP", dest as "Destination IP", answer as "DNS Answer" anslen as "Answer Length" record_type as "DNS Record Type" firstTime as "First Time" lastTime as "Last Time" count as Count | table "Source IP" "Destination IP" "DNS Answer" "DNS Record Type" "Answer Length" Count "First Time" "Last Time"

Update - ***removed eval statement for answer_length since this is already a k/v pair in Corelight data. Added > evaluation to datamodel filer to improve performance.***
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as lastTime from datamodel=Network_Resolution where DNS.record_type=TXT AND "DNS.answer_length">100 by DNS.src DNS.dest DNS.answer DNS.record_type DNS.answer_length | `drop_dm_object_name("DNS")` | `security_content_ctime(firstTime)` | `security_content_ctime(lastTime)` | rename src as "Source IP", dest as "Destination IP", answer as "DNS Answer" record_type as "DNS Record Type" firstTime as "First Time" lastTime as "Last Time" count as Count | table "Source IP" "Destination IP" "DNS Answer" "DNS Record Type" answer_length Count "First Time" "Last Time"

[Detection of DNS Tunnels]

Default search:
| tstats `security_content_summariesonly` dc("DNS.query") as count from datamodel=Network_Resolution where nodename=DNS "DNS.message_type"="QUERY" NOT (`cim_corporate_web_domain_search("DNS.query")`) NOT "DNS.query"="*.in-addr.arpa" NOT ("DNS.src_category"="svc_infra_dns" OR "DNS.src_category"="svc_infra_webproxy" OR "DNS.src_category"="svc_infra_email*") by "DNS.src","DNS.query" **| rename "DNS.src" as src "DNS.query" as message | eval length=len(message) | stats sum(length) as length by src | append [tstats `security_content_summariesonly` dc("DNS.answer") as count from datamodel=Network_Resolution where nodename=DNS "DNS.message_type"="QUERY" NOT (`cim_corporate_web_domain_search("DNS.query")`) NOT "DNS.query"="*.in-addr.arpa" NOT ("DNS.src_category"="svc_infra_dns" OR "DNS.src_category"="svc_infra_webproxy" OR "DNS.src_category"="svc_infra_email*") by "DNS.src","DNS.answer" | rename "DNS.src" as src "DNS.answer" as message | eval message=if(message=="unknown","", message) | eval length=len(message) | stats sum(length) as length by src]** | stats sum(length) as length by src | where length > 10000

Update – ***removed calculation of answer_length since this is already a Corelight k/v pair. Removed rename statements as they are not useful in this search. Removed append statement as it provided no additional data or filtering.***
| tstats `security_content_summariesonly` dc("DNS.query") as count from datamodel=Network_Resolution where nodename=DNS NOT (`cim_corporate_web_domain_search("DNS.query")`) NOT "DNS.query"="*.in-addr.arpa" NOT ("DNS.src_category"="svc_infra_dns" OR "DNS.src_category"="svc_infra_webproxy" OR "DNS.src_category"="svc_infra_email*") by "DNS.src","DNS.dest","DNS.query","DNS.answer_length" | stats sum("DNS.answer_length") as length by "DNS.src","DNS.dest" | where length>10000

[DNS Query Length With High Standard Deviation]

Default search:

| tstats `security_content_summariesonly` count from datamodel=Network_Resolution by DNS.query DNS.record_type | `drop_dm_object_name("DNS")` **| eval query_length = len(query)** | table query query_length record_type count | eventstats stdev(query_length) AS stdev avg(query_length) AS avg p50(query_length) AS p50| where query_length>(avg+stdev*2) | eval z_score=(query_length-avg)/stdev

Update – ***removed eval statement for query_length since this is already a k/v pair in Corelight data***

| tstats `security_content_summariesonly` count from datamodel=Network_Resolution by DNS.query DNS.record_type DNS.query_length | `drop_dm_object_name("DNS")` | table query query_length record_type count | eventstats stdev(query_length) AS stdev avg(query_length) AS avg p50(query_length) AS p50 | where query_length>(avg+stdev*2) | eval z_score=(query_length-avg)/stdev

[DNS Query Requests Resolved by Unauthorized DNS Servers]

Default search:

| tstats `security_content_summariesonly` count from datamodel=Network_Resolution where DNS.dest_category != dns_server AND DNS.src_category != dns_server by DNS.src DNS.dest | `drop_dm_object_name("DNS")` | `unauthorized_dns_servers_filter`

Update – no modification required

[DNS record changed]

Default search:

| inputlookup discovered_dns_records.csv | rename answer as discovered_answer | join domain[|tstats `security_content_summariesonly` count values(DNS.record_type) as type, values(DNS.answer) as current_answer values(DNS.src) as src from datamodel=Network_Resolution where **DNS.message_type=RESPONSE** DNS.answer!="unknown" DNS.answer!="" by DNS.query | rename DNS.query as query | where query!="unknown" | rex field=query "(?<domain>\w+\.\w+?)(?:$|/)"] | makemv delim=" " answer | makemv delim=" " type | sort -count | table count,src,domain,type,query,current_answer,discovered_answer | makemv current_answer | mvexpand current_answer | makemv discovered_answer | eval n=mvfind(discovered_answer, current_answer) | where isnull(n) | `dns_record_changed_output_filter`

Update – ***removed message_type=RESPONSE because all Corelight DNS logs contain a response (answer)***

| inputlookup discovered_dns_records.csv | rename answer as discovered_answer | join domain[|tstats `security_content_summariesonly` count values(DNS.record_type) as type, values(DNS.answer) as current_answer values(DNS.src) as src from datamodel=Network_Resolution where DNS.answer!="unknown" DNS.answer!="" by DNS.query | rename DNS.query as query | where query!="unknown" | rex field=query "(?<domain>\w+\.\w+?)(?:$|/)"] | makemv delim=" " answer | makemv delim=" " type | sort -count | table count,src,domain,type,query,current_answer,discovered_answer | makemv current_answer |

mvexpand current_answer | makemv discovered_answer | eval n=mvfind(discovered_answer, current_answer) | where isnull(n) | `dns_record_changed_output_filter`

[Excessive DNS Failures]

Default search:

| tstats `security_content_summariesonly` count values("DNS.query") as queries from datamodel=Network_Resolution where nodename=DNS "DNS.reply_code"!="No Error" "DNS.reply_code"!="NoError" DNS.reply_code!="unknown" NOT "DNS.query"="*.arpa" **"DNS.query"="*.*"** by "DNS.src","DNS.query"| `drop_dm_object_name("DNS")`| lookup cim_corporate_web_domain_lookup domain as query OUTPUT domain| where isnull(domain)| lookup update=true alexa_lookup_by_str domain as query OUTPUT rank| where isnull(rank)| stats sum(count) as count mode(queries) as queries by src| `get_asset(src)`| where count>50

Update – ***removed query requirement as all Corelight DNS logs have a query value.***

| tstats `security_content_summariesonly` count values("DNS.query") as queries from datamodel=Network_Resolution where nodename=DNS "DNS.reply_code"!="No Error" "DNS.reply_code"!="NoError" DNS.reply_code!="unknown" NOT "DNS.query"="*.arpa" by "DNS.src","DNS.query"| `drop_dm_object_name("DNS")`| lookup cim_corporate_web_domain_lookup domain as query OUTPUT domain| where isnull(domain)| lookup update=true alexa_lookup_by_str domain as query OUTPUT rank| where isnull(rank)| stats sum(count) as count mode(queries) as queries by src| `get_asset(src)`| where count>50

[Large Volume of DNS ANY Queries]

Default search:

| tstats `security_content_summariesonly` count from datamodel=Network_Resolution where nodename=DNS **"DNS.message_type"="QUERY"** "DNS.record_type"="ANY" by "DNS.dest" | `drop_dm_object_name("DNS")` | where count>200

Update – ***removed DNS.message_type="QUERY". "DNS.record_type="ANY" is not a value ever seen in a DNS log to our knowledge. The value in a query for ANY or ALL will be a "*". We have created a field called dns_any which will result in a true value for any query of "*" or a false value for any query without a "*".***

| tstats allow_old_summaries=true count from datamodel=Network_Resolution where nodename=DNS "DNS.dns_any"="true" by DNS.dest | `drop_dm_object_name("DNS")` | where count>75

[Monitor DNS For Brand Abuse]

Default search:

| tstats `security_content_summariesonly` values(DNS.answer) as IPs min(_time) as firstTime from datamodel=Network_Resolution by DNS.src, DNS.query | `drop_dm_object_name("DNS")` | `security_content_ctime(firstTime)`| `brand_abuse_dns`

Update – no modification required

[Detect hosts connecting to dynamic domain providers]

Default search:

| tstats `security_content_summariesonly` count values(DNS.answer) as answer min(_time) as firstTime from datamodel=Network_Resolution by DNS.src, DNS.query | `drop_dm_object_name("DNS")` | `security_content_ctime(firstTime)` | `dynamic_dns_providers`

Update – no modification required

New correlation searches

[Detect DNS on non-standard port]

| tstats `security_content_summariesonly` count from datamodel=Network_Traffic where All_Traffic.dest_port!="53" AND All_Traffic.service="dns" by All_Traffic.src_ip, All_Traffic.dest_ip, All_Traffic.dest_port | where count > 5

[Detect DNS connections to external DNS devices]

| tstats `security_content_summariesonly` count from datamodel=Network_Traffic where All_Traffic.service="dns" AND All_Traffic.direction="outbound" AND All_Traffic.src_category!="dns_server" by All_Traffic.src_ip, All_Traffic.dest_ip, All_Traffic.dest_port | where count>5

Conclusion

Corelight data simplifies and empowers DNS monitoring with Splunk Enterprise Security.

Built-in and custom correlation searches, key indicators, saved searches, and analytics stories will all function well using Corelight DNS logs. Protocol Intelligence dashboards will be fully usable based on Corelight DNS logs.